Equivalence of SQL Queries In Presence of Embedded Dependencies

Rada Chirkova
Department of Computer Science
NC State University, Raleigh, NC 27695, USA
chirkova@csc.ncsu.edu

Michael R. Genesereth
Department of Computer Science
Stanford University, Stanford, CA 94305, USA
genesereth@stanford.edu

ABSTRACT

We consider the problem of finding equivalent minimal-size reformulations of SQL queries in presence of embedded dependencies [1]. Our focus is on select-project-join (SPJ) queries with equality comparisons, also known as safe conjunctive *(CQ)* queries, possibly with grouping and aggregation. For SPJ queries, the semantics of the SQL standard treat query answers as *multisets* (a.k.a. *bags*), whereas the stored relations may be treated either as sets, which is called *bag-set semantics* for query evaluation, or as bags, which is called *bag semantics*. (Under *set semantics,* both query answers and stored relations are treated as sets.)

In the context of the above Query-Reformulation Problem, we develop a comprehensive framework for equivalence of CQ queries under bag and bag-set semantics in presence of embedded dependencies, and make a number of conceptual and technical contributions. Specifically, we develop equivalence tests for CQ queries in presence of arbitrary sets of embedded dependencies under bag and bag-set semantics, under the condition that chase [10] under set semantics *(set-chase)* on the inputs terminates. We also present equivalence tests for *aggregate* CQ queries in presence of embedded dependencies. We use our equivalence tests to develop sound and complete (whenever set-chase on the inputs terminates) algorithms for solving instances of the Query-Reformulation Problem with CQ queries under each of bag and bag-set semantics, as well as for instances of the problem with aggregate queries.

Some of our results are of independent interest. In particular, it is known that constraints that force some relations to be sets on all instances of a given database schema arise naturally in the context of sound (i.e., correct) chase [9] under bag semantics. We develop a formal framework for defining such constraints as embedded dependencies, provided that *row (tuple) IDs,* commonly used in commercial database-management systems, are defined for the respective relations.

We also extend the condition of [4] for bag equivalence of CQ queries, to those cases where some relations are set valued in all instances of the given schema. Our proof of this nontrivial result includes reasoning involving bag (non)containment. In particular, we provide an original proof (adapted to our context) of the result of [4] that CQ query Q_1 is bag contained in CQ query Q_2 only if, for each predicate used in Q_1, Q_2 has at least as many subgoals with this predicate as Q_1 does.

Our contributions are clearly applicable beyond the Query-Reformulation Problem considered in this paper. Specifically, the results of this paper can be used in developing algorithms for rewriting CQ queries and queries in more expressive languages (e.g., including grouping and aggregation, or arithmetic comparisons) using views in presence of embedded dependencies, under bag or bag-set semantics for query evaluation.

This text contains corrections to Sections 2.4 and 4 of [5].

1. INTRODUCTION

Query containment and equivalence were recognized fairly early as fundamental problems in database query evaluation and optimization. The reason is, for conjunctive queries (*CQ queries*) — a broad class of frequently used queries, whose expressive power is equivalent to that of select-project-join queries in relational algebra — query equivalence can be used as a tool in query optimization. Specifically, to find a more efficient *and* answer-preserving formulation of a given CQ query, it is enough to "try all ways" of arriving at a "shorter" query formulation, by removing query subgoals, in a process called query minimization [2]. A subgoal-removal step succeeds only if equivalence (via containment) of the "original" and "shorter" query formulations can be ensured. The equivalence test of [2] for CQ queries is known to be NP complete, whereas equivalence of general relational queries is undecidable.

In recent years, there has been renewed interest in the study of query containment and equivalence, because of their close relationship to the problem of answering queries using views [17]. In particular, the problem of rewriting relational queries equivalently using views has been the subject of extensive rigorous investigations. Please see [11, 17, 21, 23] for discussions of the state of the art and of the numerous practical applications of the problem. A test for equivalence of a CQ query to its candidate CQ rewriting in terms of CQ views uses an equivalent transformation of the rewriting to its CQ *expansion,* which (informally speaking) replaces references to views in the rewriting by their definitions [17, 23]. Then the equivalence test succeeds if and only if the expansion of the rewriting is equivalent, via the equivalence test of [2], to the input query.

Some of the investigations discussed in [11, 17, 21, 23] focused on view-based query rewriting in presence of integrity constraints (also called *dependencies*, see [1] for an overview). For a given query, accounting for the dependencies that hold on the database schema may increase the number of equivalent rewritings of the query using the given views. As a result, for a particular quality metric on the rewritings being generated, one may achieve better quality of the outputs of the rewriting generator, with obvious practical advantages. Similarly, accounting for the existing dependencies in reformulating queries in a query optimizer could result in a larger space of equivalent reformulations. For an illustration, please see Example 4.1 in this paper.

In the settings of query reformulation and view-based query rewriting in presence of dependencies, Deutsch and colleagues have developed an algorithm, called Chase and Backchase *(C&B*, see [11]) that, for a given CQ query, outputs equivalent minimal-size CQ reformulations or rewritings of the query. The technical restriction on the algorithm is the requirement that the process of "chasing" (see [1] for an overview) the input query under the available dependencies terminate in finite time. Intuitively, the point of the chase in C&B is to use the available dependencies to derive a new query formulation, which can be used to check "dependency-aware" equivalence of the query to any candidate reformulation or rewriting by using any known *dependency-free* equivalence test (e.g., that of [2] for CQ queries). Under the above restriction, the C&B algorithm is sound and complete for CQ queries, views, and rewritings/reformulations in presence of *embedded dependencies*, which are known to be sufficiently expressive to specify all usual integrity constraints, such as keys, foreign keys, inclusion, join, and multivalued dependencies [10].

The above guarantees of C&B hold under *set semantics* for query evaluation, where both the database (stored) relations and query answers are treated as sets. Query answering and rewriting in the set-semantics setting have been studied extensively in the database-theory literature. At the same time, the set semantics are *not* the default query-evaluation semantics in database systems in practice. Specifically, the expected semantics of query evaluation in the standard query language SQL [15] are *bag-set semantics.* That is, whenever a query does not use the `DISTINCT` keyword, then query answers are treated in the SQL standard as multisets (i.e., sets with duplicates, also called *bags*), whereas the database relations are assumed to be sets.

Arguably, the default semantics of SQL are the *bag semantics,* where both query answers and stored relations are permitted to be bags. Indeed, by the SQL standard stored relations are bags, rather than sets, whenever the `PRIMARY KEY` and `UNIQUE` clauses (which arise from the best practices but are not required in the SQL standard) are not part of the `CREATE TABLE` statement. Using bag semantics in evaluating SQL queries becomes imperative in presence of materialized views [17], where the definitions of some of the views may not have included the `DISTINCT` keyword, even assuming that all the original stored relations are required to be sets.

The problem of developing tests for equivalence of CQ queries under bag and bag-set semantics was solved by Chaudhuri and Vardi in [4]. The bag-set-semantics test of [4] is also used in testing equivalence of queries with grouping and aggregation [8, 22]. At the same time, developing tests for equivalence of CQ queries under bag or bag-set semantics in presence of embedded dependencies has been an open problem until now. To the best of our knowledge, the only efforts in this direction have been undertaken by Deutsch in [9] and by Cohen in [6], please see Section 7 for a more detailed discussion. Neither effort has resulted in equivalence tests for queries in presence of arbitrary sets of embedded dependencies, which may serve as an indication that the problem of developing tests for equivalence of CQ queries under bag or bag-set semantics in presence of embedded dependencies is not trivial.

Our contributions.

We consider the problem of finding equivalent minimal-size reformulations of SQL queries in presence of embedded dependencies, with a focus on select-project-join queries with equality comparisons, also known as safe CQ queries, possibly with grouping and aggregation. To construct algorithms that would solve instances of this Query-Reformulation Problem (specified in Section 3), we develop a comprehensive framework for equivalence of CQ queries under bag and bag-set semantics in presence of embedded dependencies, and make a number of conceptual and technical contributions. Specifically:

- We formulate sufficient and necessary conditions for correctness *(soundness)* of chase for CQ queries and arbitrary sets of embedded dependencies under bag and bag-set semantics, see Section 4.
- It has been shown [9] that constraints that force some relations to be sets on all instances of a given database schema arise naturally in the context of sound chase under bag semantics. We develop a formal framework for defining such constraints as embedded dependencies, provided that *row (tuple) IDs* (commonly used in commercial database-management systems) are defined for the respective relations. See Section 4 and Appendix C.
- We extend the condition of [4] for bag equivalence of CQ queries, to those cases where some relations are set valued in all instances of the given schema, see Section 4. Our proof of this nontrivial result includes reasoning involving bag (non)containment. In particular, we provide an original proof (adapted to our context) of the result of [4] that CQ query Q_1 is bag contained in CQ query Q_2 only if, for each predicate used in Q_1, Q_2 has at least as many subgoals with this predicate as Q_1 does.
- We show that the result Q_n of sound chase of a CQ query Q using a finite set Σ of embedded dependencies is unique under each of bag and bag-set semantics, whenever set-chase of Q using Σ terminates. We also provide a constructive characterization of the maximal subset of Σ that is satisfied by the canonical database for Q_n. See Section 5.
- We provide equivalence tests for CQ queries in presence of embedded dependencies under bag and bag-set semantics, see Section 6.1.
- We present equivalence tests for CQ queries *with grouping and aggregation* in presence of embedded dependencies, see Section 6.2.
- Finally, we develop sound and complete (whenever *set*-chase on the inputs terminates) algorithms for solving instances of the Query-Reformulation Problem with CQ queries under each of bag and bag-set semantics, as well as instances of the problem with aggregate queries, see Section 6.3.

Our contributions are clearly applicable beyond the Query-Reformulation Problem of Section 3. Specifically, the results of this paper can be used in developing algorithms for rewriting CQ queries and queries in more expressive languages (e.g., including grouping and aggregation, or including arithmetic comparisons [19]) using views in presence of embedded dependencies, under bag or bag-set semantics for query evaluation. Among other directions, our results could help solve the problem of reformulation for XQueries with bag semantics on XML data. Such queries can be explicitly written using the keyword `unordered`, see [9] for a discussion.

2. PRELIMINARIES

2.1 The Basics

A database schema $\mathcal{D}$ is a finite set of relation symbols and their arities. A database (instance) D over $\mathcal{D}$ has one finite relation for every relation symbol in $\mathcal{D}$, of the same arity. A relation is, in general, *bag valued*; that is, it is a bag (also called *multiset*) of tuples. A bag can be thought of as a set of elements (the *core-set* of the bag) with multiplicities attached to each element. We say that a relation is *set valued* if its cardinality coincides with the cardinality of its core-set. A database instance is, in general, *bag valued.* We say that a (bag-valued) database instance is *set valued* if all its relations are set valued.

A *conjunctive query (CQ query)* Q over a schema $\mathcal{D}$ is an expression of the form $Q(\bar{X})\ :-\ \phi(\bar{X},\bar{Y})$, where $\phi(\bar{X},\bar{Y})$ is a nonempty conjunction of atomic formulas (i.e., relational atoms, also called *subgoals*) over $\mathcal{D}$. We follow the usual notation and separate the atoms in a query by commas. We call $Q(\bar{X})$ the *head* and $\phi(\bar{X},\bar{Y})$ the *body.* We use a notation such as $\bar{X}$ for a vector of k variables and constants $X_1,\ldots,X_k$ (not necessarily distinct). Every variable in the head must appear in the body (i.e., Q must be *safe*). The set of variables in $\bar{Y}$ is assumed to be existentially quantified.

Given two conjunctions $\phi(\bar{U})$ and $\psi(\bar{V})$ of atomic formulas, a *homomorphism* from $\phi(\bar{U})$ to $\psi(\bar{V})$ is a mapping h from the set of variables and constants in $\bar{U}$ to the set of variables and constants in $\bar{V}$ such that (1) $h(c)=c$ for each constant c, and (2) for every atom $r(U_1,\ldots,U_n)$ of ϕ, $r(h(U_1),\ldots,h(U_n))$ is in ψ. Given two CQ queries $Q_1(\bar{X})\ :-\ \phi(\bar{X},\bar{Y})$ and $Q_2(\bar{X}')\ :-\ \psi(\bar{X}',\bar{Y}')$, a *containment mapping* from Q_1 to Q_2 is a homomorphism h from $\phi(\bar{X},\bar{Y})$ to $\psi(\bar{X}',\bar{Y}')$ such that $h(\bar{X})=\bar{X}'$.

For a conjunction $\phi(\bar{U})$ of atomic formulas, an *assignment* γ for $\phi(\bar{U})$ is a mapping of the variables of $\phi(\bar{U})$ to constants, and of the constants of $\phi(\bar{U})$ to themselves. We use a notation such as $\gamma(\bar{X})$ to denote tuple $(\gamma(X_1),\ldots,\gamma(X_k))$. Let relation P_i in database D correspond to predicate p_i. Then we say that *atom $p_i(\bar{X})$ is satisfied by assignment γ w.r.t. database D* if there exists tuple $t\in P_i$ in D such that $t=\gamma(\bar{X})$. Note that the satisfying assignment γ is a homomorphism from $p_i(\bar{X})$ to the ground atom $p_i(\gamma(\bar{X}))$ representing tuple t in P_i. Both the tuple-based definition of satisfaction and its homomorphism formulation are naturally extended to define satisfaction of conjunctions of atoms.

Query evaluation under set semantics. For a CQ query $Q(\bar{X})\ :-\ \phi(\bar{X},\bar{Y})$ and for a database D, suppose that there exists an assignment γ for the body $\phi(\bar{X},\bar{Y})$ of Q, such that $\phi(\bar{X},\bar{Y})$ is satisfied by γ w.r.t. D. Then we say that Q *returns a tuple* $t=\gamma(\bar{X})$ *on* D. Further, the *answer $Q(D,S)$ to Q on a set-valued database D under set semantics for query evaluation* is the set of all tuples that Q returns on D.

Query equivalence under set semantics. Query Q_1 *is contained in query* Q_2 *under set semantics* (*set-contained,* denoted $Q_1\sqsubseteq_S Q_2$) if $Q_1(D,S)\subseteq Q_2(D,S)$ for every set-valued database D. Query Q_1 *is equivalent to query* Q_2 *under set semantics* (*set-equivalent,* denoted $Q_1\equiv_S Q_2$) if $Q_1\sqsubseteq_S Q_2$ and $Q_2\sqsubseteq_S Q_1$. A classical result [2] states that a necessary and sufficient condition for the set-containment $Q_1\sqsubseteq_S Q_2$, for CQ queries Q_1 and Q_2, is the existence of a containment mapping from Q_2 to Q_1. This result forms the basis for a sound and complete test for set-equivalence of CQ queries, by definition of set-equivalence.

Canonical database. Every CQ query Q can be regarded as a symbolic database $D^{(Q)}$. $D^{(Q)}$ is defined as the result of turning each subgoal $p_i(\ldots)$ of Q into a tuple in the relation P_i that corresponds to predicate p_i. The procedure is to keep each constant in the body of Q, and to replace consistently each variable in the body of Q by a distinct constant different from all constants in Q. The tuples that correspond to the resulting ground atoms are the only tuples in the *canonical database* $D^{(Q)}$ for Q, which is unique up to isomorphism.

2.2 Bag and Bag-Set Semantics

In this section we provide definitions for query evaluation under bag and bag-set semantics. Our definitions are consistent with the semantics of evaluating CQ queries in the SQL standard (see, e.g., [15]), as well as with the corresponding definitions in [4, 18].

Query evaluation under bag-set semantics. Consider a CQ query $Q(\bar{X})\ :-\ \phi(\bar{X},\bar{Y})$. The *answer $Q(D,BS)$ to Q on a set-valued database D under bag-set semantics for query evaluation* is the bag of all tuples that Q returns on D. That is, for each assignment γ for the body $\phi(\bar{X},\bar{Y})$ of Q, such that $\phi(\bar{X},\bar{Y})$ is satisfied by γ w.r.t. D, γ contributes to the bag $Q(D,BS)$ a *distinct* tuple $t=\gamma(\bar{X})$, such that Q returns t on D w.r.t. γ. (I.e., whenever Q returns t_1 on D w.r.t. γ_1 and Q returns a copy t_2 of t_1 on D w.r.t. $\gamma_2\neq\gamma_1$, then each of t_1 and t_2 is a separate element of the bag $Q(D,BS)$.)

Query evaluation under bag semantics. For a CQ query Q, the *answer $Q(D,B)$ to Q on a bag-valued database D under bag semantics for query evaluation* is a bag of tuples computed as follows. Suppose Q is

$$Q(\bar{X})\ :-\ p_1(\bar{X}_1),p_2(\bar{X}_2),\ldots,p_n(\bar{X}_n).$$

Consider the vector $p_1,\ldots,p_n$ of predicates (not necessarily distinct) occurring in the body of Q, and let $P_1,\ldots,P_n$ be the vector of relations in D such that each p_i corresponds to relation P_i. Whenever two subgoals $p_i(\ldots)$ and $p_j(\ldots)$ of Q, with $i\neq j$, have the same predicate, P_i and P_j refer to the same relation in D.

Let γ be an assignment for the body of Q, such that the body of Q is satisfied by γ w.r.t. D. Assignment γ maps each subgoal $p_i(\bar{X}_i)$ of Q into a tuple $t^{(i)}$ in relation P_i. For each $i\in\{1,\ldots,n\}$, let m_i be the number of occurrences of tuple $t^{(i)}$ in the bag P_i. (I.e., $m_i>0$ is the multiplicity associated with the (unique copy of) tuple $t^{(i)}$ in the core-set of P_i.) Then each distinct γ contributes exactly $\Pi_{i=1}^{n}m_i$ copies of tuple $t=\gamma(\bar{X})$ to the bag $Q(D,B)$. (Recall that $\bar{X}$ is the vector of variables and constants in the head of Q.) Further, the bag $Q(D,B)$ has no other tuples.

2.3 Equivalence Tests for CQ Queries

This subsection outlines equivalence tests for CQ queries, for the cases of bag and bag-set semantics. The classical equivalence test [2] for CQ queries for the case of set semantics is described in Section 2.1.

Query equivalence under bag and bag-set semantics. Query Q_1 *is equivalent to query* Q_2 *under bag semantics* (*bag-equivalent,* denoted $Q_1 \equiv_B Q_2$) if for all bag-valued databases D it holds that $Q_1(D, B)$ and $Q_2(D, B)$ are the same bags. Query Q_1 *is equivalent to query* Q_2 *under bag-set semantics* (*bag-set-equivalent,* $Q_1 \equiv_{BS} Q_2$) if for all set-valued databases D it holds that $Q_1(D, BS)$ and $Q_2(D, BS)$ are the same bags.

PROPOSITION 2.1. *[4] Given two CQ queries Q_1 and Q_2, $Q_1 \equiv_B Q_2$ implies $Q_1 \equiv_{BS} Q_2$, and $Q_1 \equiv_{BS} Q_2$ implies $Q_1 \equiv_S Q_2$.* □

For bag and bag-set semantics, the following conditions are known for CQ query equivalence. (Query Q_c is a *canonical representation* of query Q if Q_c is the result of removing all duplicate atoms from Q.)

THEOREM 2.1. *[4] Let Q and Q' be CQ queries. Then (1) $Q \equiv_B Q'$ iff Q and Q' are isomorphic. (2) $Q \equiv_{BS} Q'$ iff $Q_c \equiv_B Q'_c$, where Q_c and Q'_c are canonical representations of Q and Q', respectively.* □

2.4 Dependencies and Chase

Embedded dependencies. We consider dependencies σ of the form

$\sigma : \phi(\bar{U}, \bar{W}) \rightarrow \exists \bar{V}\ \psi(\bar{U}, \bar{V})$

where ϕ and ψ are conjunctions of atoms, which may include equations. Such dependencies, called *embedded dependencies,* are sufficiently expressive to specify all usual integrity constraints, such as keys, foreign keys, inclusion, and join dependencies [10]. If ψ consists only of equations, then σ is an *equality-generating dependency (egd).* If ψ consists only of relational atoms, then σ is a *tuple-generating dependency (tgd).* Every set Σ of embedded dependencies is equivalent to a set of tgds and egds [1]. We write $D \models \Sigma$ if database D satisfies all the dependencies in Σ. All sets Σ we refer to are finite.

Query containment and equivalence under dependencies. We say that query Q is *set-equivalent to* query P *under a set of dependencies* Σ, denoted $Q \equiv_{\Sigma,S} P$, if for every set-valued database D such that $D \models \Sigma$ we have $Q(D, S) = P(D, S)$. The definition of *set containment under dependencies,* denoted $\sqsubseteq_{\Sigma,S}$, as well as the definitions of *bag equivalence and bag-set equivalence under dependencies* (denoted by $\equiv_{\Sigma,B}$ and $\equiv_{\Sigma,BS}$, respectively), are analogous modifications of the respective definitions for the dependency-free setting, see Sections 2.1 and 2.3.

Chase. Assume a CQ query $Q(\bar{X})\ :-\ \xi(\bar{X}, \bar{Y})$ and a tgd σ of the form $\phi(\bar{U}, \bar{W}) \rightarrow \exists \bar{V}\ \psi(\bar{U}, \bar{V})$. Assume w.l.o.g. that Q has none of the variables $\bar{V}$. The *chase of Q with σ is applicable* if there is a homomorphism h from ϕ to ξ and if, moreover, h cannot be extended to a homomorphism h' from $\phi \wedge \psi$ to ξ. In that case, a *chase step* of Q with σ and h is a rewrite of Q into $Q'(\bar{X})\ :-\ \xi(\bar{X}, \bar{Y}) \wedge \psi(h(\bar{U}), \bar{V})$.

We now define a chase step with an egd. Assume a CQ query Q as before and an egd e of the form $\phi(\bar{U}) \rightarrow U_1 = U_2$. The *chase of Q with e is applicable* if there is a homomorphism h from ϕ to ξ such that $h(U_1) \neq h(U_2)$ and at least one of $h(U_1)$ and $h(U_2)$ is a variable; assume w.l.o.g. that $h(U_1)$ is a variable. Then a *chase step* of Q with e and h is a rewrite of Q into a query that results from replacing all occurrences of $h(U_1)$ in Q by $h(U_2)$.

A Σ-*chase sequence* C (or just *chase sequence,* if Σ is clear from the context) is a sequence of CQ queries $Q_0, Q_1, \ldots$ such that every query Q_{i+1} ($i \geq 0$) in C is obtained from Q_i by a chase step $Q_i \Rightarrow^{\sigma} Q_{i+1}$ using a dependency $\sigma \in \Sigma$. A chase sequence $Q = Q_0, Q_1, \ldots, Q_n$ is *terminating under set semantics* if $D^{(Q_n)} \models \Sigma$, where $D^{(Q_n)}$ is the canonical database for Q_n. In this case we say that $(Q)_{\Sigma,S} = Q_n$ is the (terminal) *result* of the chase. Chase of CQ queries under set semantics is known to terminate in finite time for a class of embedded dependencies called *weakly acyclic dependencies*, see [14] and references therein. Under set semantics, all chase results for a given CQ query are equivalent in the absence of dependencies [10].

The following result is immediate from [1, 9, 10].

THEOREM 2.2. *Given CQ queries Q_1, Q_2 and set Σ of embedded dependencies. Then $Q_1 \equiv_{\Sigma,S} Q_2$ iff $(Q_1)_{\Sigma,S} \equiv_S (Q_2)_{\Sigma,S}$ in the absence of dependencies.* □

2.5 Queries with Grouping and Aggregation

We assume that the data we want to aggregate are real numbers, $\mathbf{R}$. If S is a set, then $\mathcal{M}(S)$ denotes the set of finite bags over S. A *k-ary aggregate function* is a function $\alpha :\ \mathcal{M}(\mathbf{R}^k)\ \rightarrow\ \mathbf{R}$ that maps bags of k-tuples of real numbers to real numbers. An *aggregate term* is an expression built up using an aggregate function over variables. Every aggregate term with k variables gives rise to a k-ary aggregate function in a natural way.

We use $\alpha(y)$ as an abstract notation for a unary aggregate term, where y is the variable in the term. Aggregate queries that we consider have (unary or 0-ary) aggregate functions *count*, *count*$(*)$, *sum*, *max*, and *min*. Note that *count* is over an argument, whereas *count*$(*)$ is the only function that we consider here that takes no argument. (There is a distinction in SQL semantics between *count* and *count*$(*)$.) In the rest of the paper, we will not refer again to the distinction between *count* and *count*$(*)$, as our results carry over.

An *aggregate query* [8, 22] is a conjunctive query augmented by an aggregate term in its head. For a query with a k-ary aggregate function α, the syntax is:

$$Q(\bar{S}, \alpha(\bar{Y})) \leftarrow A(\bar{S}, \bar{Y}, \bar{Z})\ . \tag{1}$$

A is a conjunction of atoms; $\alpha(\bar{Y})$ is a k-ary aggregate term; $\bar{S}$ are the *grouping attributes* of Q; none of the variables in $\bar{Y}$ appears in $\bar{S}$. Finally, Q is *safe*: all variables in $\bar{S}$ and $\bar{Y}$ occur in A. We consider queries with unary aggregate functions *sum*, *count*, *max*, and *min*. With each aggregate query Q as in Equation (1), we associate its CQ *core* $\breve{Q}$: $\breve{Q}(\bar{S}, \bar{Y}) \leftarrow A(\bar{S}, \bar{Y}, \bar{Z})$.

We define the semantics of an aggregate query as follows: Let D be a set-valued database and Q an aggregate query as in Equation (1). When Q is applied on D it yields a relation $Q(D)$ defined by the following three steps: First, we compute the bag $\mathbf{B} = \breve{Q}(D, BS)$ on D. We then form equivalence classes in $\mathbf{B}$: Two tuples belong to the same equivalence class if they agree on the values of all the grouping arguments of Q. This is the *grouping* step. The third step is *aggregation*; it associates with each equivalence class a value that is the aggregate function computed on a bag that contains all values of the input argument(s) of the aggregated attribute(s) in this class. For each class, it returns one tuple, which contains the values of the grouping arguments of Q and the computed aggregated value.

In general, queries with different aggregate functions may be equivalent [8]. We follow the approach of [8, 22] by considering equivalence between queries with the same lists of head arguments, called *compatible queries.*

DEFINITION 2.1. (**Equivalence of compatible aggregate queries [22]**) *For queries $Q(\bar{X}, \alpha(\bar{Y})) \leftarrow A(\bar{S})$ and $Q'(\bar{X}, \alpha(\bar{Y})) \leftarrow A'(\bar{S}')$, $Q \equiv Q'$ if $Q(D) = Q'(D)$ for every database D.* □

We say that two compatible aggregate queries Q and Q' are *equivalent in presence of a set of dependencies* Σ, $Q \equiv_\Sigma Q'$, if $Q(D) = Q'(D)$ for every database $D \models \Sigma$.

THEOREM 2.3. *[8, 22] (1) Equivalence of sum- and of count-queries can be reduced to bag-set equivalence of their cores. (2) Equivalence of max- and of min-queries can be reduced to set equivalence of their cores.* □

3. PROBLEM STATEMENT

In this section we use the following notation: Let X be the semantics for query evaluation, with values S, B, and BS, for set, bag, or bag-set semantics, respectively. Let $\mathcal{L}_1$ and $\mathcal{L}_2$ be two query languages. Let Σ be a finite set of dependencies on database schema $\mathcal{D}$.

We use the notion of Σ-minimality [11], defined as follows. (Intuitively, reformulation R of query Q is not Σ-minimal if at least one egd in Σ is applicable to R.)

DEFINITION 3.1. (**Minimality under dependencies [11]**) *A CQ query Q is* Σ-minimal *if there are no queries S_1, S_2 where S_1 is obtained from Q by replacing zero or more variables with other variables of Q, and S_2 by dropping at least one atom from S_1 such that S_1 and S_2 remain equivalent to Q under Σ.* □

We extend this definition to Σ-minimality of CQ queries with grouping and aggregation, which is defined as Σ-minimality of the (unaggregated) core of the query, see Section 2.5 for the relevant definitions.

A general statement of **the Query-Reformulation Problem** that we consider in this paper is as follows: The *problem input* is $(\mathcal{D}, X, Q, \Sigma, \mathcal{L}_2)$, where query Q is defined on database schema $\mathcal{D}$ in language $\mathcal{L}_1$. A *solution* to the Query-Reformulation Problem, for a problem input $(\mathcal{D}, X, Q, \Sigma, \mathcal{L}_2)$, is a query Q' defined in language $\mathcal{L}_2$ on $\mathcal{D}$, such that $Q' \equiv_{\Sigma,X} Q$.

In this paper we consider the Query-Reformulation Problem in presence of embedded dependencies, and focus on (1) the *CQ class* of the problem, where each of $\mathcal{L}_1$ and $\mathcal{L}_2$ is the language of CQ queries, and on (2) the *CQ-aggregate class* (see Section 6.3), where each of $\mathcal{L}_1$ and $\mathcal{L}_2$ is the language of CQ queries with grouping and aggregation, using aggregate functions *sum*, *max*, *min*, and *count*; we refer to this query language as *CQ-aggregate.* For both classes, we consider only Σ*-minimal solutions* of the Query-Reformulation Problem.

4. SOUND CHASE UNDER BAG AND BAG-SET SEMANTICS

In this section we show that under bag and bag-set semantics, it is incorrect to enforce the set-semantics condition of $D^{(Q_n)} \models \Sigma$ (Section 2.4) on the terminal chase result Q_n of query Q under dependencies Σ. The problem is that under this condition, chase may yield a result Q_n that is *not* equivalent to the original query Q in presence of Σ. That is, soundness of chase, understood as $Q_n \equiv_{\Sigma,B} Q$ or $Q_n \equiv_{\Sigma,BS} Q$, may *not* hold. We then formulate sufficient and necessary conditions for soundness of chase for CQ queries and embedded dependencies under bag and bag-set semantics.

In this section we also show that constraints that force certain relations to be sets on all instances of a given database schema can be defined as egds, provided that *row (tuple) IDs* are defined for the respective relations. Finally, we extend the condition of Theorem 2.1 for bag equivalence of CQ queries, to those cases where some relations are required to be set valued in all instances of the given schema. Such requirements can be defined as our set-enforcing egds.

4.1 Motivating Example

Let us conjecture that maybe an analog of Theorem 2.2 (Section 2.4) holds for the case of bag semantics. (In this section we discuss in detail the case of bag semantics only; analogous reasoning is valid for the case of bag-set semantics.) That is, maybe $Q_1 \equiv_{\Sigma,B} Q_2$ if and only if $(Q_1)_{\Sigma,S} \equiv_B (Q_2)_{\Sigma,S}$ in the absence of dependencies, for a given pair of CQ queries Q_1 and Q_2 and for a given set Σ of embedded dependencies. (We obtain our conjecture by replacing the symbols $\equiv_{\Sigma,S}$ and $\equiv_S$ in Theorem 2.2 by the *bag*-semantics versions of these symbols.)

Now consider the C&B algorithm by Deutsch and colleagues [11]. Under set semantics for query evaluation and given a CQ query Q, C&B outputs all equivalent Σ-minimal conjunctive reformulations of Q in presence of the given embedded dependencies Σ (i.e., C&B is *sound and complete*), whenever chase of Q under Σ terminates in finite time. See Appendix A for the details on C&B.

If our conjecture is valid, then a straightforward modification of C&B gives us a procedure for solving instances in the CQ class of the Query-Reformulation Problem for bag semantics.[1] The only difference between the original C&B and its proposed modification would be the test for *bag*, rather than *set*, equivalence (see Theorem 2.1) between the universal plan $(Q)_{\Sigma,S}$ of C&B for the input query Q and dependencies Σ, and the terminal result of chasing a candidate reformulation. (These terms are defined in Appendix A.) By extension from C&B, our algorithm would be sound and complete for all problem instances where the universal plan for Q could be computed in finite time.

Unfortunately, this naive extension of C&B would not be sound for bag semantics (or for bag-set semantics, in the version of C&B using the bag-set equivalence test of Thm. 2.1). We highlight the problems in an example.

EXAMPLE 4.1. *On database schema $\mathcal{D} = \{P, R, S, T, U\}$, consider a set Σ that includes four tgds:*

$\sigma_1 : p(X, Y) \rightarrow s(X, Z) \wedge t(X, V, W)$
$\sigma_2 : p(X, Y) \rightarrow t(X, Y, W)$
$\sigma_3 : p(X, Y) \rightarrow r(X)$
$\sigma_4 : p(X, Y) \rightarrow u(X, Z) \wedge t(X, Y, W)$

Suppose Σ also includes dependencies enforcing the following constraints: (1) Relations S and T (but not

[1]An analogous extension of C&B would work for instances $(\mathcal{D}, BS, Q, \Sigma, CQ)$, i.e., under bag-set semantics.

R or U) are set valued in all instances of $\mathcal{D}$; call these constraints σ_5 and σ_6, respectively. (These dependencies are relevant to the bag-semantics case. Under bag-set or set semantics, all relations in all instances of $\mathcal{D}$ are set valued by definition.) Please see Section 4.2 for an approach to expressing such constraints using egds. (2) The first attribute of S is the key of S (egd σ_7), and the first two attributes of T are the key of T (egd σ_8), see Appendix B for the definition of keys.

Consider CQ queries Q_1 through Q_4, defined as

$$\begin{aligned}
Q_1(X) &:- p(X,Y), t(X,Y,W), s(X,Z), r(X), u(X,U).\\
Q_2(X) &:- p(X,Y), t(X,Y,W), s(X,Z), r(X).\\
Q_3(X) &:- p(X,Y), t(X,Y,W), s(X,Z).\\
Q_4(X) &:- p(X,Y).
\end{aligned}$$

(We disregard queries Q_2 and Q_3 for the moment.)

We can show that $Q_1 \equiv_{\Sigma,S} Q_4$. Thus, Q_1 is a reformulation of Q_4 in presence of Σ under set semantics. At the same time, by [2] Q_1 and Q_4 are not equivalent under set semantics in the absence of dependencies.

Our naive modification of C&B would return a reformulation Q_1 of query Q_4. Indeed, each of $(Q_1)_{\Sigma,S}$ and $(Q_4)_{\Sigma,S}$ is isomorphic to Q_1, thus by Theorem 2.1 we have that $(Q_1)_{\Sigma,S} \equiv_B (Q_4)_{\Sigma,S}$.

However, even though $(Q_1)_{\Sigma,S} \equiv_B (Q_4)_{\Sigma,S}$, it is not true that $Q_1 \equiv_{\Sigma,B} Q_4$. The counterexample is a bag-valued database D, $D \models \Sigma$, with relations $P = \{\{(1,2)\}\}$, $R = \{\{(1)\}\}$, $S = \{\{(1,3)\}\}$, $T = \{\{(1,2,4)\}\}$, and $U = \{\{(1,5),(1,6)\}\}$. On the database D, the answer to Q_4 under bag semantics is $Q_4(D,B) = \{\{(1)\}\}$, whereas $Q_1(D,B) = (Q_1)_{\Sigma,S}(D,B) = (Q_4)_{\Sigma,S}(D,B) = \{\{(1),(1)\}\}$. From the fact that $Q_1(D,B)$ and $Q_4(D,B)$ are not the same bags, *we conclude that $Q_1 \not\equiv_{\Sigma,B} Q_4$.*

The same database D (which is set *valued) would disprove $Q_1 \equiv_{\Sigma,BS} Q_4$ (i.e., equivalence of Q_1 and Q_4 under Σ and* bag-set *semantics), even though it is true by Theorem 2.1 that $(Q_1)_{\Sigma,S} \equiv_{BS} (Q_4)_{\Sigma,S}$.* □

4.2 Sound Chase Steps

The problem highlighted in Example 4.1 is unsoundness of *set*-semantics chase when applied to query Q_4 under bag or bag-set semantics. To rectify this problem, that is to make chase sound under these semantics, we modify the definitions of chase steps.

Given a CQ query Q and a set of embedded dependencies Σ, let Q' be the result of applying to query Q a dependency $\sigma \in \Sigma$. We say that *the chase step $Q \Rightarrow^{\sigma}_{B} Q'$ is sound under bag semantics* [9] *($Q \Rightarrow^{\sigma}_{BS} Q'$ is sound under bag-set semantics, respectively)* if it holds that $Q \equiv_{\Sigma,B} Q'$ (that $Q \equiv_{\Sigma,BS} Q'$, respectively). By extension of the above definitions, all chase steps under embedded dependencies are sound under *set* semantics. The definitions of sound chase steps are naturally extended to those of *sound chase sequences* under each semantics. We say that a *chase result Q_n is sound w.r.t. (Q,Σ) under bag semantics (under bag-set semantics, respectively)* whenever there exists a Σ-chase sequence C that starts with the input query Q and ends with Q_n, and such that all chase steps in C are sound under bag semantics (under bag-set semantics, respectively).

4.2.1 Regularized Assignment-Fixing Tgds

Toward ensuring soundness of chase under bag and bag-set semantics, we will define key-based chase using tgds, see Section 4.2.3. For our definition we will need the technical notions of "regularized tgds" and "assignment-fixing tgds", which we formally define and characterize in this subsection.

Regularized tgds.

Consider a tgd $\sigma : \phi(\bar{X},\bar{Y}) \rightarrow \exists \bar{Z}\ \psi(\bar{X},\bar{Z})$ whose right-hand side ψ has at least two relational atoms. Let ψ_a and ψ_b be a partition of ψ (where ψ is viewed as set of relational atoms) in σ into two disjoint nonempty sets, that is $\psi_a \neq \emptyset$, $\psi_b \neq \emptyset$, $\psi_a \cap \psi_b = \emptyset$, and $\psi_a \cup \psi_b = \psi$. Let $\bar{A}$ be all the variables in ψ_a, and let $\bar{B}$ be all the variables in ψ_b. We call ψ_a and ψ_b a *nonshared partition of ψ* in σ whenever $\bar{A} \cap \bar{B} \subseteq \bar{X}$. (Recall that all the variables in $\bar{X}$ are universally quantified in σ.) In case where ψ_a and ψ_b are two disjoint nonempty sets such that $\psi_a \cup \psi_b = \psi$ and $\bar{A} \cap \bar{B} \cap \bar{Z} \neq \emptyset$, we call ψ_a and ψ_b a *shared partition of ψ* in σ.

Definition 4.1. *(***Regularized tgd, regularized set of embedded dependencies***) A tgd $\sigma : \phi \rightarrow \psi$ is a* regularized tgd *if there exists no nonshared partition of the set of relational atoms of ψ into two disjoint nonempty sets.*[2] *We say that a finite set Σ of embedded dependencies is a* regularized set of (embedded) dependencies *if each tgd in Σ is regularized.* □

Sets $\{u(X,Z)\}$ and $\{t(X,Y,W)\}$ comprise a nonshared partition of the right-hand side of tgd σ_4 in Example 4.1; therefore, the tgd σ_4 is not regularized. For a tgd σ_1 in Example 4.2, where $\sigma_1 :\ p(X,Y) \rightarrow \exists Z\ \exists W\ r(X,Z) \wedge s(Z,W)$, sets $\{r(X,Z)\}$ and $\{s(Z,W)\}$ comprise a shared partition of the right-hand side of σ_1, because an existential variable Z of σ_1 is present in both elements of the partition. This tgd is regularized by Definition 4.1. The set Σ in Example 4.6 is a regularized set of dependencies.

Consider a tgd $\sigma : \phi \rightarrow \psi$ that is not regularized by Definition 4.1. The process of *regularizing σ* is the process of constructing from σ a set $\Sigma_\sigma = \{\sigma_1,\ldots,\sigma_k\}$ of tgds, where $k \geq 2$ and such that for each tgd σ_i in Σ_σ, (i) the left-hand side of σ_i is the left-hand side ϕ of σ; (ii) the right-hand side of σ_i is a nonempty set of atoms $\psi_i \subseteq \psi$ (recall that ψ is the right-hand side of σ), with all the existential variables (of ψ) in ψ_i marked as such in σ_i; (iii) σ_i is regularized by Definition 4.1; and (iv) $\cup_{i=1}^{k}\psi_i = \psi$. It is easy to see that given a non-regularized tgd σ, the recursive algorithm of finding nonshared partitions of the right-hand side of σ (a) regularizes σ correctly, (b) results in a unique set Σ_σ, and (c) has the complexity $O(m^2\ log\ m)$, where m is the number of relational atoms in the right-hand side of σ. (The idea of the algorithm is to (1) give a unique ID $id(a_\psi)$ to each relational atom a_ψ of ψ, to then (2) associate with each $id(a_\psi)$ the set of all those variables of a_ψ that are existentially quantified in σ, and to then (3) recursively sort all the ids, each time by one fixed variable in their associated variable lists, and to either start a new nonshared partition using the sorted list, or to add atoms to an existing nonshared partition, again using the sorted list.) We call Σ_σ the *regularized set* of σ.

Now given a finite set Σ of arbitrary embedded egds and tgds, we *regularize* Σ by regularizing each tgd in

[2]Trivially, every tgd whose right-hand side has exactly one atom is a regularized tgd.

Σ as described above. We say that Σ' is a *regularized version of* Σ if (i) for each egd σ in Σ, Σ' also has σ, (ii) for each tgd σ in Σ, Σ' has the regularized set of σ and, finally, (iii) Σ' has no other dependencies. For each Σ as above, it is easy to see that Σ' is regularized by Definition 4.1 and is unique.

The following results, in Proposition 4.1, are immediate from Definition 4.1 and from the constructions in this subsection.

PROPOSITION 4.1. *For a finite set Σ of embedded egds and tgds defined on schema $\mathcal{D}$, let Σ' be the regularized version of Σ. Then*

- *For every bag-valued database D with schema $\mathcal{D}$, $D \models \Sigma$ iff $D \models \Sigma'$; and*
- *For every CQ query Q defined on $\mathcal{D}$, chase of Q under set semantics in presence of Σ terminates in finite time iff chase of Q under set semantics in presence of Σ' terminates in finite time, and $(Q)_{\Sigma,S} \equiv_S (Q)_{\Sigma',S}$ provided both chase results exist.* □

Assignment-fixing tgds.

In the remainder of the paper, whenever we refer to a set of embedded dependencies, we assume that we are discussing (or using) its regularized *version.* We now define assignment-fixing tgds. The idea is to be able to determine easily which tgds ensure sound chase steps under each of bag and bag-set semantics. The intuition is as follows. Suppose chase with tgd σ is applicable to a CQ query Q as defined in Section 2.4 (i.e., assuming set semantics), but we are looking at the implications of applying the chase under bag or bag-set semantics rather than under set semantics. Suppose further that the right-hand side of σ has existential variables. Then we would like to add subgoals to Q, that is to perform on Q the chase step $Q \Rightarrow^{\sigma} Q'$, exactly in those cases where each consistent assignment to all body variables of Q, w.r.t. any (arbitrary) database D that satisfies the input dependencies, can be extended to *one and only one* consistent assignment to all body variables of Q' w.r.t. D. Otherwise Q would not be equivalent to Q' in presence of σ and under the chosen query-evaluation semantics. It turns out that the characterization we are seeking is, in general, *query dependent.* (See Examples 4.3 and 5.1.)

We now formalize this intuition of prohibiting, in chase, "incorrect" multiplicity of the answer to the given query in presence of the given dependencies, under bag or bag-set semantics. Consider a CQ query $Q(\bar{A}) \ :- \ \zeta(\bar{A}, \bar{B})$, and a regularized tgd $\sigma : \phi(\bar{X}, \bar{Y}) \rightarrow \exists \bar{Z} \ \psi(\bar{X}, \bar{Z})$ that has at least one existential variable, that is $\bar{Z}$ is not empty. Suppose that chase of Q with σ is applicable, using a homomorphism h from ϕ to ζ. We come up with a substitution θ of all existential variables $\bar{Z}$ in the right-hand side ψ of the tgd σ, such that θ replaces each variable in $\bar{Z}$ by a fresh variable that is not used in any capacity (i.e., neither universally nor existentially quantified) in σ or in ζ. (Observe that θ always exists.) We use h and θ to define for Q and σ an *associated test query* $Q^{\sigma,h,\theta}$:

$$Q^{\sigma,h,\theta}(\bar{A}) \ :- \ \zeta(\bar{A}, \bar{B}) \wedge \psi(h(\bar{X}), \bar{Z}) \wedge \psi(h(\bar{X}), \theta(\bar{Z})) \ . \tag{2}$$

Observe that for any pair (θ_1, θ_2) of substitutions that satisfy the conditions on θ above, Q^{σ,h,θ_1} and Q^{σ,h,θ_2} are isomorphic. Hence $Q^{\sigma,h,\theta}$ is unique up to isomorphism w.r.t. θ, and we choose one arbitrary θ for $Q^{\sigma,h,\theta}$ in the remainder of the paper.

We now treat the case where σ has no existential variables. In this case θ is trivially empty, $\theta = \emptyset$, and we define the associated test query $Q^{\sigma,h,\emptyset}$ for Q, σ, and h as above as:

$$Q^{\sigma,h,\emptyset}(\bar{A}) : -\zeta(\bar{A}, \bar{B}) \wedge \psi(h(\bar{X}), \bar{Z}) \ . \tag{3}$$

That is, $Q^{\sigma,h,\emptyset}$ is the result of applying to the query Q a chase step using σ, as defined in Section 2.4 in this paper.

We stress again that Equation 3 is defined only for those cases where σ has no existential variables. However, Equation 3 can be obtained from Equation 2 by setting $\theta = \emptyset$ and by removing duplicate subgoals from the body of the query in Equation 2. Therefore, in what follows we adopt Equation 2 as the definition of the associated test query for Q and σ regardless of whether σ has existential variables.

DEFINITION 4.2. *(**Associated test query**) Given a CQ query Q and a regularized tgd σ such that chase using σ is applicable to Q using homomorphism h, the* associated test query *for Q, σ, and h is as shown in Equation 2.* □

We now define assignment-fixing tgds, which enable sound chase steps under each of bag and bag-set semantics, under an extra condition under bag semantics that all the subgoals being added in the chase step correspond to set-valued relations. We first ensure correctness of the definition of assignment-fixing tgds, by making a straightforward observation.

PROPOSITION 4.2. *Given a CQ query Q and a finite set Σ of tgds and egds, and for a regularized tgd*[3] *$\sigma \in \Sigma$ such that chase using σ applies to Q with a homomorphism h. Then the terminal chase result $(Q^{\sigma,h,\theta})_{\Sigma,S}$ exists whenever $(Q)_{\Sigma,S}$ exists.* □

PROOF. (Sketch.) Trivial for the case where σ has no existential variables. For the remaining case, the proof is by contradiction. Suppose that $(Q^{\sigma,h,\theta})_{\Sigma,S}$ does not exist, that is, the body of $(Q^{\sigma,h,\theta})_{\Sigma,S}$ has an infinite number of relational subgoals, using an infinite number of variable names. We then show that the body of $(Q)_{\Sigma,S}$ also has an infinite number of relational subgoals (using an infinite number of variable names), and thus arrive at the desired contradiction. The procedure is to apply to Q all the chase steps that are applicable to $Q^{\sigma,h,\theta}$. Specifically, for each chase step S that applies on $Q^{\sigma,h,\theta}$ using a homomorphism μ, we apply the same chase step to *the result Q' of chase step on Q using σ and the h of $Q^{\sigma,h,\theta}$.* In each S we use the homomorphism that is a composition of μ with a homomorphism that results from putting together the identity mapping (on some of ths subgoals) and θ^{-1}, for the θ used in defining $Q^{\sigma,h,\theta}$. (By definition, θ is injective and thus θ^{-1} exists.)

[3]Recall that we assume throughout the paper that Σ is the regularized version of any given set of tgds and egds.

Observe that this "simulation" on Q' of the infinite chase on $Q^{\sigma,h,\theta}$ cannot collapse the infinite number of variables in $(Q^{\sigma,h,\theta})_{\Sigma,S}$ into a finite number of variables (and thus into a *finite* number of subgoals) in the "simulation result". The reason is, the language of embedded dependencies cannot specify the instruction "generate a new variable name, using the right-hand side of the tgd in question, *only* if some variable names are not the same in the left-hand side of the tgd in question". Q.E.D. $\square$

We are finally ready to define assignment-fixing gds.

DEFINITION 4.3. *(***Assignment-fixing tgd***) Given a CQ query Q and a finite set Σ of tgds and egds such that $(Q)_{\Sigma,S}$ exists, let $\sigma \in \Sigma$ be a regularized tgd with existential variables $Z_1, \ldots, Z_k$, $k \geq 0$, such that chase of Q with σ is applicable, with associated test query $Q^{\sigma,h,\theta}$. Then σ is* an assignment-fixing tgd w.r.t. Q and h *if $(Q^{\sigma,h,\theta})_{\Sigma,S}$ has at most one of Z_i and $\theta(Z_i)$ for each $i \in \{1, \ldots, k\}$. Further, σ is an* assignment-fixing tgd w.r.t. Q *if σ is an assignment-fixing tgd w.r.t. Q and some homomorphism h.* $\square$

PROPOSITION 4.3. *In the setting of Definition 4.3, whenever σ is a full tgd (i.e., tgd without existential variables), then σ is an assignment-fixing tgd w.r.t. all CQ queries Q such that chase using σ is applicable to Q and such that $(Q)_\Sigma$ exists.* $\square$

Consider two illustrations of the determination whether a tgd with existential variables is assignment fixing w.r.t. a given CQ query. Example 4.2 is a positive example, in that it establishes a tgd as assignment fixing, whereas Example 4.3 is a negative example.

EXAMPLE 4.2. *On database schema $\mathcal{D} = \{P, R, S\}$, consider a regularized set of embedded dependencies $\Sigma = \{\sigma_1, \sigma_2, \sigma_3\}$, where σ_1 is a tgd,*

$$\sigma_1 : \ p(X,Y) \rightarrow \exists Z \ \exists W \ r(X,Z) \wedge s(Z,W),$$

egd σ_2 establishes the first attribute of R as its superkey, and, finally, egd σ_3 is as follows:

$$\sigma_3 : \ r(X,Y) \wedge s(Y,T) \wedge r(X,Z) \wedge s(Z,W) \rightarrow T = W.$$

Let CQ query Q be $Q(X) \ :- \ p(X,Y)$. Chase using σ_1 is applicable to Q, using homomorphism $h = \{X \rightarrow X, Y \rightarrow Y\}$. For the query

$$\begin{aligned} Q^{\sigma_1,h,\theta}(X) \ :- \ & p(X,Y), r(X,Z), s(Z,W), \\ & r(X,Z_1), s(Z_1,W_1) \ . \end{aligned}$$

constructed using $\theta = \{Z \rightarrow Z_1, W \rightarrow W_1\}$, we have

$$(Q^{\sigma_1,h,\theta})_{\Sigma,S}(X) \ :- \ p(X,Y), r(X,Z), s(Z,W) \ .$$

Thus, σ_1 is an assignment-fixing tgd w.r.t. Q, because the body of $(Q^{\sigma_1,h,\theta})_{\Sigma,S}(X)$ has only one of Z and Z_1 and only one of W and W_1. $\square$

EXAMPLE 4.3. *Using the database schema and dependency σ_2 of Example 4.2, we replace σ_1 of that example with a regularized tgd σ_4:*

$$\sigma_4 : \ p(X,Y) \rightarrow \exists Z, W, T \ r(X,Z) \wedge s(Z,W) \wedge s(X,T).$$

We also replace σ_3 of the example with egd σ_5, and add an egd σ_6:

$$\sigma_5 : \ r(X,Z) \wedge s(Z,W) \wedge s(X,T) \rightarrow W = T.$$
$$\sigma_6 : \ p(X,Y) \wedge r(A,X) \wedge s(X,T) \rightarrow X = T.$$

We denote by Σ' the set of dependencies $\{\sigma_2, \sigma_4, \sigma_5, \sigma_6\}$.

Consider again query $Q(X) \ :- \ p(X,Y)$. Chase using σ_4 is applicable to Q, using the identity homomorphism h. For the query

$$\begin{aligned} Q^{\sigma_4,h,\theta}(X) \ :- \ & p(X,Y), r(X,Z), s(Z,W), s(X,T) \\ & r(X,Z_1), s(Z_1,W_1), s(X,T_1) \ . \end{aligned}$$

constructed using $\theta = \{Z \rightarrow Z_1, W \rightarrow W_1, T \rightarrow T_1\}$, we have

$$\begin{aligned} (Q^{\sigma_4,h,\theta})_{\Sigma',S}(X) \ :- \ & p(X,Y), r(X,Z), \\ & s(Z,W), s(X,W), s(Z,W_1), s(X,W_1) \ . \end{aligned}$$

Thus, σ_4 is not *an assignment-fixing tgd w.r.t. Q by definition, because the body of $(Q^{\sigma_4,h,\theta})_{\Sigma',S}(X)$ has both of W and W_1.* $\square$

4.2.2 Motivation for Regularized Assignment-Fixing Tgds

One may wonder whether the notions introduced in Section 4.2.1 are justified. In this subsection we illustrate that whenever a non-regularized tgds or a tgd that is not assignment fixing is used in chase step $Q \Rightarrow^\sigma Q'$, then the chase result Q' may be nonequivalent to Q under bag or bag-set semantics.

Examples 4.4 through 4.6 establish the need for regularized tgds and for the (traditional) definition of the chase step for tgds, see Section 2.4 in this paper. Example 4.7 shows an unsound chase step using a regularized tgd that is not assignment fixing w.r.t. the query. Finally, Example 4.8 demonstrates a sound chase step using a regularized assignment-fixing tgd, and illustrates how the notion of assignment-fixing tgds is strictly more general than that of key-based dependencies (see Definition 5.1).

EXAMPLE 4.4. *Consider Example 4.1, where tgd σ_4 is not key based in presence of the set Σ of embedded dependencies in the example, by the definition of [9], see Definition 5.1. For the reader convenience, we provide here the tgd σ_4 and query Q_4 of Example 4.1.*

$\sigma_4 : p(X,Y) \rightarrow u(X,Z) \wedge t(X,Y,W)$
$Q_4(X) \ :- \ p(X,Y).$

Now consider the result of removing from Σ the tgd σ_2 of Example 4.1; we denote by Σ' the set $\Sigma' = \Sigma - \{\sigma_2\}$. In presence of Σ', the tgd σ_4 is still not key based. However, if we refrain from applying σ_4 to Q_4 in chase under bag or bag-set semantics, then we will miss the rewriting Q_3 (of Example 4.1) of Q_4. Indeed, by the results of this paper it holds that $Q_3 \equiv_{\Sigma',B} Q_4$ and that $Q_3 \equiv_{\Sigma',BS} Q_4$. $\square$

Observe that tgd σ_4 in Example 4.4 is not regularized, see Definition 4.1. We miss an equivalent rewriting of the input query Q_4 by refraining from applying the tgd. Consider now Example 4.5, where we do apply the nonregularized tgd σ_4 in its entirety to the query Q_4. However, instead of the query Q_3, which is equivalent to Q_4 in presence of Σ' under each of bag and bag-set semantics, we obtain a formulation of Q_4 that is not equivalent to Q_4 (in presence of Σ') under either semantics.

EXAMPLE 4.5. *Consider the query Q_4 and set Σ' of dependencies in Example 4.4. We now attempt to find the rewriting Q_3 (of Example 4.1) that we failed to obtain in Example 4.4.*

$Q_3(X)\ :-\ p(X,Y), t(X,Y,W), s(X,Z).$

To find the rewriting Q_3, specifically to obtain its T-subgoal, we apply the nonregularized dependency σ_4 to the query Q_4. We denote by Q_4' the result of the application:

$Q_4'(X)\ :-\ p(X,Y), t(X,Y,W), u(X,Z).$

Recall from Example 4.1 that in presence of Σ', relation U does not have superkeys other than the set of all its attributes. Using this information, we construct a database D that is a counterexample to equivalence of Q_4 and Q_4' in presence of Σ' and under bag-set semantics. (Thus, by definition, D is also a counterexample to the equivalence of the queries in presence of Σ' and under bag semantics as well).

Let $D = \{P(1,2), T(1,2,3), U(1,4), U(1,5)\}$. (Observe that D is a set-valued database and that $D \models \Sigma'$.) On database D, $Q_4(D,BS) = \{\!\{\ (1)\ \}\!\}$, whereas $Q_4'(D,BS) = \{\!\{\ (1),(1)\ \}\!\}$. □

Note 1 on Example 4.5. The problem with applying σ_4 to query Q_4 in the example is that σ_4 is not regularized. The regularized set for σ_4 is $\{\sigma_4', \sigma_4''\}$, where

$\sigma_4' : p(X,Y) \rightarrow t(X,Y,W)$
$\sigma_4'' : p(X,Y) \rightarrow u(X,Z)$

Observe that tgd σ_4' is assignment fixing in presence of (the egds in) Σ' (of Example 4.4), whereas σ_4'' is not. Thus, σ_4'' cannot be applied in sound chase of Q_4 using Σ' under bag or bag-set semantics, by our main results of this section. Using the regularized version of Σ' (this version also replaces σ_1 of Example 4.1 with its regularized set), we can perform sound chase Q_4 to obtain the above query Q_3, which is equivalent to Q_4 in presence of Σ' under each of bag and bag-set semantics (with the usual restriction of set-valued relations in the case of bag semantics).

We now examine the modified definition of chase, see Section 2.4 of [5]. Indeed, using that definition we obtain correctly the terminal chase results of the query Q_4 in Example 4.1, even though not all input tgds are regularized, see Examples 4.1 and 5.1 of [5]. However, as we see in the next example, using the modified definition of chase does not result in sound chase (under bag or bag-set semantics) for all problem inputs.

EXAMPLE 4.6. *Consider query Q and set $\Sigma = \{\nu_1, \nu_2\}$ of dependencies, where*

$Q(X)\ :-\ p(X,Y), s(X,Z)$
$\nu_1 : p(X,Y) \rightarrow \exists Z\ s(X,Z) \wedge t(Z,Y)$
$\nu_2 : t(X,Y) \wedge t(Z,Y) \rightarrow X = Z$

Observe that ν_1 is a regularized tgd and is also assignment fixing, w.r.t. Q, by our definitions in this section. We now apply modified chase as defined in Section 2.4 of [5] and obtain query Q':

$Q'(X)\ :-\ p(X,Y), s(X,Z), t(Z,Y).$

We show nonequivalence of Q to Q' in presence of Σ under each of bag and bag-set semantics, by constructing a database D that is a counterexample to either equivalence. Indeed, let $D = \{P(1,2), S(1,1), S(1,3), T(3,2)\}$. (Observe that D is a set-valued database and that $D \models \Sigma$.) On database D, $Q(D,BS) = \{\!\{\ (1),(1)\ \}\!\}$, whereas $Q'(D,BS) = \{\!\{\ (1)\ \}\!\}$. □

Note on Example 4.6. The application of ν_1 to Q in the example is sound by the (incorrect) definition of key-based chase steps in [5]. Still, the application of the regularized and assignment-fixing tgd ν_1 using the *modified* definition of the chase step does result in unsound chase as shown in Example 4.6.

We now show an example of using a regularized but *not* assignment fixing tgd in a (traditional) chase step, see Section 2.4 in this paper for the definition.

EXAMPLE 4.7. *Recall the database schema $\mathcal{D} = \{P, R, S\}$ and dependencies $\Sigma' = \{\sigma_2,\ \sigma_4, \sigma_5\}$ of Example 4.3.*

$\sigma_2 : r(X,Y) \wedge r(X,Z) \rightarrow Y = Z\ .$
$\sigma_4 : p(X,Y) \rightarrow \exists Z, W, T\ r(X,Z) \wedge s(Z,W) \wedge s(X,T)\ .$
$\sigma_5 : r(X,Z) \wedge s(Z,W) \wedge s(X,T) \rightarrow W = T\ .$

Recall that σ_4 is regularized but not assignment fixing w.r.t. query $Q(X)\ :-\ p(X,Y)$; see Example 4.3 for the details. We apply the chase step using tgd σ_4 to Q, to obtain the result Q'':

$Q(X)\ :-\ p(X,Y)\ .$
$Q''(X)\ :-\ p(X,Y), r(X,Z), s(Z,W), s(X,T)\ .$

To construct a counterexample to equivalence of Q and Q'' in presence of Σ', under each of bag and bag-set semantics, we use the query $(Q^{\sigma_4,h,\theta})_{\Sigma',S}$ of Example 4.3. Specifically, we use as a counterexample the canonical database, call it D, of $(Q^{\sigma_4,h,\theta})_{\Sigma',S}$; we have that D is set valued and that $D \models \Sigma'$ by definition of the query $(Q^{\sigma_4,h,\theta})_{\Sigma',S}$.

Consider the database $D = \{P(1,2), R(1,3), S(1,4), S(1,5), S(3,4), S(3,5)\}$. (Recall that the canonical database of a CQ query is isomorphic up to choice of constants.) We have that $Q(D,BS) = \{\!\{\ (1)\ \}\!\}$, whereas $Q'(D,BS) = \{\!\{\ (1),(1),(1),(1)\ \}\!\}$. □

By our main results in this section, for the Q, Σ, and ν_1 of Example 4.6, the application of ν_1 to Q (using the traditional definition of chase steps using tgds, see Section 2.4 in this paper) is sound in presence of Σ under each of bag and bag-set semantics (provided that for the case of bag semantics, both S and T are set-valued relations in all instances of $\{P, S, T\}$). Example 4.8 shows the chase step.

EXAMPLE 4.8. *Consider the query Q and set $\Sigma = \{\nu_1,\ \nu_2\}$ of dependencies of Example 4.6. Recall that ν_1 is a regularized tgd and is also assignment fixing w.r.t. Q in presence of the egds of Σ. We now apply (traditional) chase as defined in Section 2.4 in this paper, to obtain query Q'':*

$Q''(X)\ :-\ p(X,Y), s(X,Z), s(X,W), t(W,Y).$

The difference from our application of ν_1 in Example 4.6 is that we now add a new S-subgoal in addition to a new T-subgoal. By the definition of chase steps using tgds, the second attribute of S must be denoted by different variables in the two S-subgoals in query Q''. □

Note on Example 4.8. Recall that ν_1 in the example is assignment fixing w.r.t. the query, and thus by our results can be applied in sound chase under bag and bag-set semantics (provided that for the case of bag semantics, both S and T are set-valued relations in all instances of $\{P, S, T\}$). At the same time, ν_1 is not key-based by the definition of [9], see Definition 5.1 in this paper. The problem is with the S-atom of ν_1, which is not key based in presence of Σ by Definition 5.1.

4.2.3 Assignment-Fixing Chase

We begin the exposition of the main results of this section by defining *assignment-fixing chase steps using tgds.*

DEFINITION 4.4. *(**Assignment-fixing chase step using tgd**) Let σ be a regularized tgd in a finite set Σ of embedded dependencies on schema $\mathcal{D}$. Consider a CQ query Q defined on $\mathcal{D}$, such that $(Q)_{\Sigma,S}$ exists and such that σ is applicable to Q. Then the chase step that applies σ to Q is an* assignment-fixing chase step using *σ whenever σ is an assignment-fixing tgd w.r.t. Q.* $\square$

We now provide necessary and sufficient conditions for soundness of chase steps under bag semantics for query evaluation.

THEOREM 4.1. *Given a CQ query Q and a set of embedded dependencies[4] Σ on schema $\mathcal{D}$. Under bag semantics, a chase step $Q \Rightarrow^{\sigma} Q'$ using $\sigma \in \Sigma$ is sound iff*

1. *$Q \Rightarrow^{\sigma} Q'$ is a (tgd) assignment-fixing chase step, and for each subgoal $s(p_{ij})$ that the chase step adds to Q, relation P_{ij} is set valued on all databases satisfying Σ; or*
2. *In $Q \Rightarrow^{\sigma} Q'$, σ is an egd; in this case, duplicates of subgoal $s(p)$ in Q' can be removed only if relation P is set valued in all instances of $\mathcal{D}$.* $\square$

In Section 4.2.3, Example 4.7 shows an unsound chase step using a regularized tgd that is not assignment fixing w.r.t. the query. Example 4.8 in Section 4.2.3 demonstrates a sound (by Theorem 4.1) chase step using a regularized assignment-fixing tgd, *provided that* both S and T are set-valued relations in all instances of the database schema used in the example. Relaxing this set-valued requirement would result in an unsound chase step using the same tgd, as is easy to demonstrate using a counterexample bag-valued database.

The requirement that certain stored relations be set valued arises naturally if one seeks soundness of bag-semantics chase, see [9]. We now show that constraints that force certain relations to be sets on all instances of a database schema can be formally defined as egds, provided that *row (tuple) IDs* are defined for the respective relations. In the common practice of using tuple IDs in database systems, each tuple in a (bag-valued) relation is assigned a unique tuple ID. Then the *set-enforcing egd* on relation P can be expressed as a functional dependency (*fd,* defined in Appendix B), which specifies that whenever two tuples of P agree on everything except the tuple IDs, then the tuples must also agree on the tuple IDs. Please see Appendix C for the details of our set-enforcing framework based on tuple IDs.

[4]Recall that we consider only finite regularized sets of dependencies throughout this paper.

We now discuss item 2 of Theorem 4.1. Given a database schema $\mathcal{D}$, suppose that for some of the relation symbols $\{P_1, \ldots, P_k\} \subseteq \mathcal{D}$ it holds that the relation for each of $P_1, \ldots, P_k$ is required to be *set* valued in all instances D over $\mathcal{D}$. For such scenarios, the bag-equivalence test of Theorem 2.1 is no longer a necessary condition for bag equivalence of CQ queries.

EXAMPLE 4.9. *By Theorem 2.1, query Q_3 of Example 4.1 is not bag equivalent to query Q_5:*

$$Q_5(X) \ :- \ p(X,Y), t(X,Y,W), s(X,Z), s(X,Z).$$

Here, the only difference between Q_3 and Q_5 is the extra copy of subgoal $s(X,Z)$ in Q_5. At the same time, Q_3 and Q_5 are *bag equivalent on all bag-valued databases where relation S is required to be a set. Please see Theorem 4.2 and Appendix D for the details.* $\square$

We now formulate the extended sufficient and necessary condition. Please see Appendix D for the proof.

THEOREM 4.2. *Let $\{P_1, \ldots, P_k\} \subseteq \mathcal{D}$ be the maximal set of relation symbols in schema $\mathcal{D}$ such that the relation for each of $P_1, \ldots, P_k$ is required to be* set *valued in all instances D over $\mathcal{D}$. Given CQ queries Q_1, Q_2 on $\mathcal{D}$, let query Q_1' (Q_2', respectively) be obtained by removing from Q_1 (from Q_2, respectively) all duplicate subgoals whose predicates correspond to $P_1, \ldots, P_k$. Then $Q_1 \equiv_B Q_2$ in the absence of all dependencies other than the set-enforcing dependencies on $P_1, \ldots, P_k$ of the schema $\mathcal{D}$ if and only if Q_1' and Q_2' are isomorphic.* $\square$

The correctness of the duplicate-removal rule of item 2 in Theorem 4.1 is immediate from Theorem 4.2.

We now spell out the necessary and sufficient conditions for soundness of chase steps under *bag-set* semantics for query evaluation.

THEOREM 4.3. *Given a CQ query Q and a set of embedded dependencies[5] Σ. Under bag-set semantics, a chase step $Q \Rightarrow^{\sigma} Q'$ using $\sigma \in \Sigma$ is sound iff*

1. *$Q \Rightarrow^{\sigma} Q'$ is a (tgd) assignment-fixing chase step; or*
2. *In $Q \Rightarrow^{\sigma} Q'$, σ is an egd.* $\square$

We use Examples 4.7 and 4.8 of Section 4.2.3 to make here the same points as for Theorem 4.1. Observe that (unlike the case of bag semantics) the set-valuedness requirement is satisfied by definition of bag-set semantics. See Example 4.1 for query Q_2 that is obtained from Q_4 by using, among other sound chase steps, a chase step involving dependency σ_3. By Theorem 4.1, σ_3 may not be used in sound chase under *bag* semantics, because relation S is not guaranteed to be set valued in all instances of the database schema of the example.

PROOF. (Theorems 4.1 and 4.3, sketch.) We outline here the correctness proof for chase steps using tgds. Please see Appendix E for the details of disproving soundness of chase steps under bag semantics whenever chase (using even regularized and assignment-fixing tgds) adds query subgoals whose associated base relations are not set valued in all instances of the given database schema.

[5]Recall that we consider only finite regularized sets of dependencies throughout this paper.

Consider a CQ query Q and a set of dependencies Σ defined on schema $\mathcal{D}$, such that $(Q)_\Sigma$ exists. Let $\sigma \in \Sigma$ be a regularized dependency such that chase using σ applies to Q (using a homomorphism h) and results in query Q'. (That is, $Q \Rightarrow^\sigma Q'$ is defined.) Further, suppose that for all subgoals that are in Q' but not in Q, the respective base relations, call them collectively $\mathcal{S} \subseteq \mathcal{D}$, are set valued in all instances of the schema $\mathcal{D}$.

Case (1): Let σ be an assignment-fixing tgd w.r.t. the query Q. We prove that on all instances D of $\mathcal{D}$ such that $D \models \Sigma$ and such that at least the relations in $\mathcal{S}$ are set valued on D, it holds that $Q(D,B) = Q'(D,B)$ and that $Q(D,BS) = Q'(D,BS)$. (Thus, the chase step $Q \Rightarrow^\sigma Q'$ is sound under the conditions of Theorems 4.1 and 4.3.)

We fix an arbitrary database D as described above. The idea of the proof is to establish a 1:1 correspondence between all the assignments satisfied by Q w.r.t. D and all the assignments satisfied by Q' w.r.t. D. As a result (and using the fact that the $\mathcal{S}$-part of the base relations in D is guaranteed to be set valued), we obtain that for each tuple $t \in Q(D,B)$, such that the multiplicity of t in $Q(D,B)$ is $m > 0$, the multiplicity of t *in* $Q'(D,B)$ is also m.

We establish the 1:1 correspondence as follows.

(i) For each assignment μ' that satisfies Q' w.r.t. D, there exists exactly one assignment μ that (a) satisfies Q w.r.t. D, and that (b) coincides with μ' on the set of body variables of Q. (Recall that σ is a tgd, and therefore the set of body variables of Q' is a superset of the set of body variables of Q.)

(ii) For each assignment μ that satisfies Q w.r.t. D, there exists at least one assignment μ' that (a) satisfies Q' w.r.t. D, and that (b) coincides with μ on the set of body variables of Q. This is immediate from the fact that $D \models \Sigma$.

(iii) From the fact that σ is assignment fixing w.r.t. Q, we obtain that for each μ as in (ii) there exists *at most* one corresponding μ' as in (ii). Indeed, suppose that for some such μ there exist at least two assignments μ'_1 and μ'_2 that satisfy the conditions of (ii). Then we show by obtaining the chase result $(Q^{\sigma,h,\theta})_{\Sigma,S}$, in Definition 4.3, that μ'_1 and μ'_2 must be identical on all databases satisfying Σ.

The observation that D is an arbitrary database satisfying the conditions above concludes the proof of $Q \equiv_{\Sigma,B} Q'$ in this case (1). Further, $Q \equiv_{\Sigma,BS} Q'$ is immediate from $Q \equiv_{\Sigma,B} Q'$.

Case (2): Let σ *not* be assignment fixing w.r.t. the query Q. We construct a set-valued database D (with schema $\mathcal{D}$) such that $D \models \Sigma$ and such that $Q(D,BS) \neq Q'(D,BS)$. (As a result, neither of $Q \equiv_{\Sigma,B} Q'$ and $Q \equiv_{\Sigma,BS} Q'$ holds, and therefore the chase step $Q \Rightarrow^\sigma Q'$ is not sound in this case under bag or bag-set semantics.)

As a counterexample database D we use the canonical database of the query $(Q^{\sigma,h,\theta})_{\Sigma,S}$, see Definition 4.3. Example 4.7 illustrates the construction.

Let ν be the satisfying (by definition of canonical databases and by definition of chase under set semantics) assignment to the head variables $\bar{X}$ of $Q^{\sigma,h,\theta}$ w.r.t. the database D. Observe that the vectors of head variables of all of Q, Q', and $Q^{\sigma,h,\theta}$ are the same by definition of $Q^{\sigma,h,\theta}$. By definition of $Q^{\sigma,h,\theta}$, there exists an extension ν_Q of ν to all the body variables of Q such that ν_Q satisfies Q w.r.t. D, and there exists an extension $\nu'_{Q'}$ of ν to all the body variables of Q' such that $\nu'_{Q'}$ satisfies Q' w.r.t. D.

We make the following observations about the answers to Q and Q' under bag-set semantics on the set-valued database D.

(i) For each assignment μ' such that $\mu'|_{\bar{X}} = \nu$ and such that μ' satisfies Q' w.r.t. D (we have shown that there exists at least one such assignment μ'), there exists exactly one assignment μ that (a) satisfies Q w.r.t. D, and that (b) coincides with μ' on the set of body variables of Q. (See (i) under case (1) of the proof.) Observe that $\mu|_{\bar{X}} = \nu$ by definition of μ.

(ii) For each assignment μ such that $\mu|_{\bar{X}} = \nu$ and such that μ satisfies Q w.r.t. D (we have shown that there exists at least one such assignment μ), there exists at least one assignment μ' that (a) satisfies Q' w.r.t. D, and that (b) coincides with μ on the set of body variables of Q. (See (ii) under case (1) of the proof.) Observe that $\mu'|_{\bar{X}} = \nu$ by definition of μ'.

(iii) On our counterexample database D, there exists at least one μ with $\mu|_{\bar{X}} = \nu$ and such that μ is a satisfying assignment w.r.t. Q and D, such that μ corresponds to *at least two distinct* satisfying assignments μ'_1 and μ'_2 w.r.t. Q' and D, where each of μ'_1 and μ'_2 coincides with μ on all the body variables of Q. Indeed, we recall that D is the canonical database of $(Q^{\sigma,h,\theta})_{\Sigma,S}$. If the distinct μ'_1 and μ'_2 as above did not exist, then chase of $Q^{\sigma,h,\theta}$ using Σ under set semantics would lead to the "elimination of the distinction between" the groups of subgoals $\psi(h(\bar{X}),\bar{Z})$ and $\psi(h(\bar{X}),\theta(\bar{Z}))$ of $Q^{\sigma,h,\theta}$, see Equation 2 and Definition 4.3, in the terminal chase result of $Q^{\sigma,h,\theta}$ using Σ. But if $\psi(h(\bar{X}),\bar{Z})$ and $\psi(h(\bar{X}),\theta(\bar{Z}))$ collapse into the same group in $(Q^{\sigma,h,\theta})_{\Sigma,S}$, then σ is an assignment-fixing tgd w.r.t. Q by Definition 4.3, which is a contradiction with our assumption.

We conclude that in Case (2), the multiplicity of the tuple $\nu(\bar{X})$ is strictly greater in $Q'(D,BS)$ than in $Q(D,BS)$ on our counterexample database D. Thus, $Q'(D,BS) \neq Q(D,BS)$. Q.E.D. $\Box$

5. UNIQUE RESULT OF SOUND CHASE

In this section we show that the result of sound chase of CQ queries using arbitrary finite sets of embedded dependencies is unique under each of bag and bag-set semantics for query evaluation. Further, we provide an algorithm for constructing, for a given CQ query Q and an arbitrary finite set of embedded depedencies Σ, the maximal subset $\Sigma^{max}_B(Q,\Sigma)$ of Σ such that $D^{(Q_n)} \models \Sigma^{max}_B(Q,\Sigma)$, where Q_n is the result of sound chase of Q under bag semantics. We also outline a version of the algorithm that works for the case of bag-set semantics.

5.1 Why Not Key-Based Tgds?

We begin the discussion by examining the question of why the definition of assignment-fixing chase steps (Definition 4.4) cannot be simplified. The intuition behind the notion of assignment-fixing chase steps is that of ensuring that in each assignment-fixing chase step $Q \Rightarrow_B^{\sigma} Q'$, using some tgd $\sigma \in \Sigma$, each tuple in the bag $Q(D, B)$ would have *the same* multiplicity in the bag $Q'(D, B)$, for each database $D \models \Sigma$, in presence of the requisite set-enforcing constraints (of Appendix C). The intuition is the same for bag-set-semantics. It appears that a simpler notion, that of *key-based tgds*, would suffice. In the definition that follows, we use the notation of Definition 4.4.

Definition 5.1. *(***Key-based tgd***) Let $\sigma : \phi(\bar{X}, \bar{Y}) \rightarrow \exists \bar{Z}\ \psi(\bar{Y}, \bar{Z})$ be a tgd on database schema $\mathcal{D}$. Then σ is a* key-based tgd *if, for each atom $p(\bar{Y}'_j, \bar{Z}'_j)$ in ψ, $\bar{Y}'_j$ is a superkey of relation P in $\mathcal{D}$ and, in addition, P is set valued on all instances of $\mathcal{D}$.* □

The notion of key-based tgds is equivalent to that of UWDs of [9]. Note that by Definition 4.4, all chase steps using key-based tgds are assignment fixing. However, the class of assignment-fixing tgds (w.r.t. the given CQ query and set of dependencies) includes not just key-based tgds, as illustrated in Example 4.8. In addition, unlike assignment-fixing chase steps specified in Definition 4.4, a key-based tgd is defined independently of the queries being chased. Deutsch [9] showed that the result of sound chase of CQ queries under bag semantics is unique up to isomorphism, provided that *all* tgds in the given set of dependencies are key based.

It turns out that the "key-basedness" constraints of Definition 5.1 on tgds are not necessary for soundness of chase under either of bag and bag-set semantics. Indeed, consider a modification of Example 4.3:

EXAMPLE 5.1. *In the setting of Example 4.3, we replace the query Q by a query $Q'(X)\ :-\ p(X, Y), r(A, X)$, and keep the set Σ' of dependencies of Example 4.3. We can show that tgd $\sigma_4 \in \Sigma'$ is assignment fixing w.r.t. Q'. Recall that σ_4 is* not *an assignment-fixing tgd w.r.t. the query Q of Example 4.3.* □

5.2 Uniqueness of Result of Sound Chase

We now show that the result of sound chase of CQ queries using *arbitrary* sets of embedded dependencies[6] is unique under bag and bag-set semantics, up to equivalence in the absence of dependencies (except for the set-enforcing dependencies under bag semantics). (Recall that throughout the paper we assume that all given sets of embedded dependencies are finite and regularized.) We give here a formulation of our result only for the case of bag semantics. The version of Theorem 5.1 for the case of bag-set semantics (formulated in Appendix G) is straightforward.

Theorem 5.1. *Given a CQ query Q and set Σ of embedded dependencies on schema $\mathcal{D}$, such that there exists a* set-*chase result $(Q)_{\Sigma,S}$ for Q and Σ. Then there exists a result $(Q)_{\Sigma,B}$ of* sound *chase for Q and Σ under* bag *semantics, unique up to isomorphism after dropping duplicate subgoals that correspond to set-valued relations in $\mathcal{D}$.[7] That is, for two sound-chase results $(Q)^{(1)}_{\Sigma,B}$ and $(Q)^{(2)}_{\Sigma,B}$ for Q and Σ, $(Q)^{(1)}_{\Sigma,B} \equiv_B (Q)^{(2)}_{\Sigma,B}$ in the absence of all dependencies other than the set-enforcing dependencies on stored relations.* □

[6]Cf. the result of [9] on uniqueness of sound bag chase for key-based tgds only; see Section 5.1 for the discussion.

By Theorem 4.1, sound bag chase adds or drops only those subgoals whose predicates correspond to relations required to be sets. Thus, it is natural to use the conditions of Theorem 4.2, rather than of Theorem 2.1, in characterizing bag equivalence of terminal chase results.

To prove Theorem 5.1, we make the following straightforward observation.

Proposition 5.1. *Given CQ query Q and embedded dependencies Σ such that there exists a* set-*chase result $(Q)_{\Sigma,S}$. Then sound chase of Q using Σ terminates in finite time under each of bag and bag-set semantics.* □

This result is immediate from Theorems 4.1 and 4.3.

The rest of the proof of Theorem 5.1 is an adaptation, to *sound* chase steps, of the proof of the fact (see [10]) that all set-chase results (when defined) for a given CQ query are equivalent in the absence of dependencies. Please see Appendix G for the details.

We now establish the complexity of sound bag and bag-set chase under weakly acyclic dependencies [14]. Intuitively, weakly acyclic dependencies cannot generate an infinite number of new variables, hence set-chase under such dependencies terminates in finite time; please see Appendix H for the definition. All sets of dependencies in examples in this paper are weakly acyclic.

Theorem 5.2. *Given a CQ query Q and set Σ of weakly acyclic embedded dependencies on schema $\mathcal{D}$. Then sound chase of Q using Σ, under each of bag and bag-set semantics, terminates in time polynomial in the size of Q and exponential in the size of Σ.* □

The upper bound is immediate from Proposition 5.1 and from the results in [1, 11, 14] for set semantics. For the lower bound, we exhibit an infinite family of pairs (Q, Σ), where the size of each of $(Q)_{\Sigma,B}$ and $(Q)_{\Sigma,BS}$ is polynomial in the size of Q and exponential in the size of Σ. Please see Appendix H for the details.

5.3 Satisfiable Dependencies Are Query Based

We now provide a constructive characterization of the result of sound chase under bag and bag-set semantics. This characterization, formulated in Theorem 5.3 for bag semantics, settles the problem of which dependencies Σ' are satisfied by the canonical database $D^{(Q_n)}$ of Q_n. Here, Q_n is the result of sound chase of CQ query Q using embedded dependencies Σ. (We assume that *set* chase of Q using Σ terminates in finite time.)

Given a CQ query Q and a set of embedded dependencies Σ, consider the canonical database $D^{(Q_n)}$ of the result $Q_n = (Q)_{\Sigma,B}$ of sound chase of Q using Σ under bag semantics. Clearly, at least some sets Σ' such that $D^{(Q_n)} \models \Sigma'$ do not coincide with the original Σ. (We refer here to the discussion in the beginning of Section 4.) For instance, in Example 4.1 the canonical database for query Q_3 does not satisfy dependency σ_4. Observe that Q_3 is the (unique, by Theorem 5.1) result of sound chase of Q_4 using Σ under bag semantics.

[7]See discussion of Theorem 4.2 in Section 4.2.

At the same time, for each pair (Q, Σ) there exists a unique maximal-size set $\Sigma_B^{max}(Q, \Sigma) \subseteq \Sigma$, such that $D^{(Q_n)} \models \Sigma_B^{max}(Q, \Sigma)$. (Appendix I has proof of Theorem 5.3 and the analogous result for bag-set semantics.)

THEOREM 5.3. **(Unique $\Sigma_B^{max}(Q, \Sigma) \subseteq \Sigma$)** *Given a CQ query Q and set Σ of embedded dependencies, such that there exists a* set-*chase result $(Q)_{\Sigma,S}$ for Q and Σ. Let Q_n be the result of sound chase for Q and Σ under* bag *semantics, with canonical database $D^{(Q_n)}$. Then there exists a unique subset $\Sigma_B^{max}(Q, \Sigma)$ of Σ, such that:*

- *$D^{(Q_n)} \models \Sigma_B^{max}(Q, \Sigma)$, and*
- *for each proper superset Σ' of $\Sigma_B^{max}(Q, \Sigma)$ such that $\Sigma' \subseteq \Sigma$, $D^{(Q_n)} \models \Sigma'$ does* not *hold.* □

It turns out that the set $\Sigma_B^{max}(Q, \Sigma)$ is the result of removing from Σ exactly those tgds σ such that the chase step $Q_n \Rightarrow_B^{\sigma} Q'$, with some CQ outcome Q', is not sound under bag semantics. This claim is immediate from the observation that for each dependency σ in Σ such that σ is applicable to Q_n, σ is *unsoundly* applicable to Q_n. See Appendix I for the details. We make the same observation about the unique set $\Sigma_{BS}^{max}(Q, \Sigma) \subseteq \Sigma$ such that $\Sigma_{BS}^{max}(Q, \Sigma)$ is the maximal set of dependencies satisfied by the canonical database of the result of sound chase of Q using Σ under *bag-set* semantics.

Not surprisingly, each of $\Sigma_B^{max}(Q, \Sigma)$ and $\Sigma_{BS}^{max}(Q, \Sigma)$ is query dependent. Recall that in Example 4.1 the canonical database of the query $Q_3 = (Q_4)_{\Sigma,B}$ does not satisfy dependency σ_4 in the set Σ given in the example. At the same time, it is easy to see that for query $Q(X) :- p(X, Y), u(X, Z)$, the canonical database of the query $(Q)_{\Sigma,B}$ *does* satisfy dependency σ_4 in the *same* set Σ.

We now establish a relationship between $\Sigma_B^{max}(Q, \Sigma)$ and $\Sigma_{BS}^{max}(Q, \Sigma)$ for a fixed pair (Q, Σ). This relationship is immediate from Theorems 4.1, 4.3, 5.3, and I.1.

PROPOSITION 5.2. *For (Q, Σ) satisfying conditions of Theorem 5.3, $\Sigma_B^{max}(Q, \Sigma) \subseteq \Sigma_{BS}^{max}(Q, \Sigma) \subseteq \Sigma$.* □

Query Q_4 and dependencies Σ of Example 4.1 can be used to show that both subset relationships can be proper: $\Sigma_B^{max}(Q, \Sigma) \subset \Sigma_{BS}^{max}(Q, \Sigma) \subset \Sigma$.

We now outline algorithm MAX-BAG-Σ-SUBSET, which accepts as inputs a CQ query Q and a finite set Σ of embedded dependencies such that $(Q)_{\Sigma,S}$ exists. The algorithm constructs the set $\Sigma_B^{max}(Q, \Sigma)$ as specified in Theorem 5.3. The counterpart of MAX-BAG-Σ-SUBSET for bag-set semantics can be found in Appendix I.

Algorithm 1: Max-Bag-Σ-Subset(Q, Σ)

```
Input  : CQ query $Q$, set $\Sigma$ of embedded dependencies
         such that chase result $(Q)_{\Sigma,S}$ exists.
Output : $\Sigma_B^{max}(Q, \Sigma) \subseteq \Sigma$ specified in Theorem 5.3.
1. $(Q)_{\Sigma,B} := soundChase(B, Q, \Sigma)$;
2. $\Sigma_B^{max}(Q, \Sigma) := \Sigma$;
3. for each $\sigma$ in $\Sigma$ do
    4. if $soundChaseStep(\sigma, B, (Q)_{\Sigma,B}) = false$ then
        5. $\Sigma_B^{max}(Q, \Sigma) := \Sigma_B^{max}(Q, \Sigma) - \{\sigma\}$;
6. return $\Sigma_B^{max}(Q, \Sigma)$;
```

The algorithm begins (line 1 of the pseudocode) by computing the result $(Q)_{\Sigma,B}$ of sound chase of Q using Σ under bag semantics (B). This result exists and is unique by Theorem 5.1. Then the algorithm removes from the set Σ all dependencies that are unsoundly applicable to $(Q)_{\Sigma,B}$, see lines 2-5 of the pseudocode. Procedure *soundChaseStep*$(\sigma, B, (Q)_{\Sigma,B})$ (line 4) returns *true* if and only if the bag-chase step using σ on $(Q)_{\Sigma,B}$ is sound by Theorem 4.1.

We obtain the following result by construction of algorithm MAX-BAG-Σ-SUBSET.

THEOREM 5.4. **(Correctness and complexity of** MAX-BAG-Σ-SUBSET**)** *Given a CQ query Q and set of embedded dependencies Σ, such that there exists a* set-*chase result $(Q)_{\Sigma,S}$ for Q and Σ. Then algorithm* MAX-BAG-Σ-SUBSET *returns in finite time the set $\Sigma_B^{max}(Q, \Sigma)$ specified in Theorem 5.3. If dependencies Σ are weakly acyclic, then the runtime of the algorithm is polynomial in the size of Q and exponential in the size of Σ.* □

6. Σ-EQUIVALENCE TESTS FOR CQ AND CQ-AGGREGATE QUERIES

We begin this section by providing equivalence tests for CQ queries in presence of embedded dependencies under bag and bag-set semantics, see Section 6.1. These results allow us to develop: (1) Equivalence tests for CQ queries *with grouping and aggregation* in presence of embedded dependencies, see Section 6.2, and (2) Sound and complete (whenever *set*-chase on the inputs terminates) algorithms for solving instances of the CQ class of the Query-Reformulation Problem under each of bag and bag-set semantics, as well as for the CQ-aggregate class of the problem, see Section 6.3. (Recall that throughout the paper we assume that all given sets of embedded dependencies are finite and regularized.)

6.1 Equivalence Tests for CQ Queries

The main results of this section for CQ queries, Theorems 6.1 and 6.2, are the analogs, for bag and bag-set semantics, of the dependency-free test of Theorem 2.2 for equivalence of CQ queries under set semantics and under embedded dependencies.

THEOREM 6.1. *Given CQ queries Q and Q', and a set of embedded dependencies Σ such that there exist* set-*chase results $(Q)_{\Sigma,S}$ for Q and $(Q')_{\Sigma,S}$ for Q'. Then $Q \equiv_{\Sigma,B} Q'$ if and only if $(Q)_{\Sigma,B} \equiv_B (Q')_{\Sigma,B}$ in the absence of all dependencies other than the set-enforcing dependencies on stored relations.*[8] □

THEOREM 6.2. *Given CQ queries Q and Q', and a set of embedded dependencies Σ such that there exist* set-*chase results $(Q)_{\Sigma,S}$ for Q and $(Q')_{\Sigma,S}$ for Q'. Then $Q \equiv_{\Sigma,BS} Q'$ if and only if $(Q)_{\Sigma,BS} \equiv_{BS} (Q')_{\Sigma,BS}$ in the absence of dependencies.* □

The proofs of Theorems 6.1 and 6.2 follow from Proposition 5.1 and from Theorem 5.1 and its analog for bag-set semantics. See Appendix J for the details.

We now formulate Proposition 6.1, which is the dependency-based version of Proposition 2.1. The proof of Proposition 6.1 can be found in Appendix K.

PROPOSITION 6.1. *For CQ queries Q and Q' and set of embedded dependencies Σ, such that there exists the* set-*chase result for each of Q and Q' using Σ. Then (1) $(Q)_{\Sigma,B} \equiv_B (Q')_{\Sigma,B}$, in the absence of all dependencies other than the set-enforcing constraints on stored relations, implies $(Q)_{\Sigma,BS} \equiv_{BS} (Q')_{\Sigma,BS}$, and (2) $(Q)_{\Sigma,BS} \equiv_{BS} (Q')_{\Sigma,BS}$ implies $(Q)_{\Sigma,S} \equiv_S (Q')_{\Sigma,S}$.* □

[8]See Theorem 4.2 and discussion of Theorem 5.1.

Observe that queries $(Q)_{\Sigma,B}$, $(Q)_{\Sigma,BS}$, and $(Q)_{\Sigma,S}$ may be distinct queries for *the same* query Q and set Σ. For an illustration, please see the chase results Q_1 through Q_3 of query Q_4 in Example 4.1.

A corollary of Proposition 6.1 establishes a *set-containment* relationship between a CQ query and the results of its sound chase under a given set of embedded dependencies. Please see Appendix K for a proof.

PROPOSITION 6.2. *For* (Q, Σ) *that satisfy conditions of Thm. 5.3,* $(Q)_{\Sigma,S} \sqsubseteq_S (Q)_{\Sigma,BS} \sqsubseteq_S (Q)_{\Sigma,B} \sqsubseteq_S Q$. $\square$

Queries Q_4, $Q_3 = (Q_4)_{\Sigma,B}$, $Q_2 = (Q_4)_{\Sigma,BS}$, and $Q_1 = (Q_4)_{\Sigma,S}$ of Example 4.1 provide an illustration.

6.2 Equivalence Tests for Aggregate Queries

We now provide dependency-free tests for equivalence of CQ queries with grouping and aggregation under embedded dependencies. The results of this subsection are immediate from Theorems 2.2, 2.3, and 6.2.

THEOREM 6.3. *Given compatible aggregate queries* Q *and* Q', *and a set of embedded dependencies* Σ *such that there exist* set-*chase results* $(\breve{Q})_{\Sigma,S}$ *for the core* $\breve{Q}$ *of* Q *and* $(\breve{Q}')_{\Sigma,S}$ *for the core* $\breve{Q}'$ *of* Q'. *Then (1) For max or min queries* Q *and* Q', $Q \equiv_\Sigma Q'$ *if and only if* $(\breve{Q})_{\Sigma,S} \equiv_S (\breve{Q}')_{\Sigma,S}$ *in the absence of dependencies. (2) For sum or count queries* Q *and* Q', $Q \equiv_\Sigma Q'$ *if and only if* $(\breve{Q})_{\Sigma,BS} \equiv_{BS} (\breve{Q}')_{\Sigma,BS}$ *in the absence of dependencies.* $\square$

6.3 Sound and Complete Reformulation of CQ and CQ-Aggregate Queries

Theorems 6.1 and 6.2 allow us to extend the algorithm C&B of [11] to (a) reformulation of CQ queries in presence of embedded dependencies under bag or bag-set semantics, and to (b) reformulation of CQ queries with grouping and aggregation in presence of embedded dependencies. Our proposed algorithm BAG-C&B returns Σ-minimal reformulations Q' of CQ query Q such that $Q' \equiv_{\Sigma,B} Q$ under the given embedded dependencies Σ. The only modifications to C&B that are required to obtain BAG-C&B are (i) to replace the set-chase procedure by the *sound bag-chase* procedure as defined in this paper, and (ii) to replace the dependency-free equivalence test of Theorem 2.2 by the test of Theorem 6.1. The algorithm BAG-SET-C&B for the case of bag-set semantics is obtained in an analogous fashion.

We have also developed algorithms that accept sets of embedded dependencies and CQ queries with grouping and aggregation: MAX-MIN-C&B accepts CQ queries with aggregate function *max* or *min*, and SUM-COUNT-C&B accepts CQ queries with aggregate function *sum* or *count*. MAX-MIN-C&B uses C&B to obtain all Σ-minimal reformulations $Q' \equiv_{\Sigma,S} \breve{Q}$ of the core $\breve{Q}$ of the input query Q, and for each such query Q' returns a query Q'' whose head is the head of Q and whose body is the body of Q'. SUM-COUNT-C&B works analogously, except that it uses BAG-SET-C&B to produce queries $Q' \equiv_{\Sigma,BS} \breve{Q}$. By Theorem 6.3, for each output Q'' of MAX-MIN-C&B or of SUM-COUNT-C&B it holds that $Q'' \equiv_\Sigma Q$ whenever set-chase of Q using Σ terminates.

All our algorithms are sound and complete whenever *set*-semantics chase of Q using Σ terminates.

THEOREM 6.4. *Given CQ query* Q *and set* Σ *of embedded dependencies such that* set *chase of* Q *under* Σ *terminates in finite time. Then* BAG-C&B *returns all* Σ*-minimal reformulations* Q' *such that* $Q' \equiv_{\Sigma,B} Q$. $\square$

The analogs of Theorem 6.4 for (a) CQ queries under bag-set semantics, and for (b) aggregate CQ queries can be found in Appendix K. All the theorems follow from the soundness and completeness of C&B of [11] (see Appendix A) and from the results of this paper.

7. RELATED WORK

Chandra and Merlin [2] developed the NP-complete containment test of two CQ queries under set semantics. This test has been used in optimization of CQ queries, as well as in developing algorithms for rewriting queries (both equivalently and nonequivalently) using views. Please see [11, 17, 21, 23] for discussions of the state of the art and of the numerous practical applications of query rewriting using views.

The problem of developing tests for equivalence of CQ queries under bag and bag-set semantics was solved by Chaudhuri and Vardi in [4]. The results on *containment* tests for CQ queries under bag semantics have proved to be more elusive. Please see Jayram and colleagues [18] for original undecidability results for containment of CQ queries with inequalities under bag semantics. The authors point out that it is not known whether the problem of bag containment for *CQ* queries is even decidable. On the other hand, the problem of containment of CQ queries under bag-set semantics reduces to the problem of containment of aggregate queries with aggregate function `count(*)`. The latter problem is solvable using the methods proposed in [7].

Studies of dependencies have been motivated by the goal of good database-schema design. See [1, 10] for overviews and references on dependencies and chase. In [9], Deutsch developed chase methods for bag-specific constraints (UWDs), and proved completeness of the view-based version of the Chase and Backchase algorithm (C&B, [11]) for mixed semantics and for set and bag dependencies, in case where all given tuple-generating dependencies are UWDs. In contrast, the algorithm in [13] is complete in presence of just functional dependencies. Algorithms that are complete in the absence of dependencies are given in [20] for set semantics, in [3] for bag semantics, and in [16] for bag-set semantics. Finally, Cohen in [6] presented an equivalence test for CQ queries in presence of inclusion dependencies,[9] for the cases of bag-set semantics and of the semantics where queries are evaluated on set-valued databases using both bag-valued and set-valued intermediate results.

8. REFERENCES

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[2] A. Chandra and P. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *STOC*, 1977.

[3] S. Chaudhuri, R. Krishnamurthy, S. Potamianos, and K. Shim. Optimizing queries with materialized views. In *ICDE*, pages 190–200, 1995.

[4] S. Chaudhuri and M. Vardi. Optimization of *real* conjunctive queries. In *PODS*, pages 59–70, 1993.

[5] R. Chirkova and M. Genesereth. Equivalence of SQL queries in presence of embedded dependencies. In *PODS*, 2009.

[6] S. Cohen. Equivalence of queries combining set and bag-set semantics. In *PODS*, pages 70–79, 2006.

[7] S. Cohen, W. Nutt, and Y. Sagiv. Containment of aggregate queries. In *ICDT*, pages 111–125, 2003.

[9] An *inclusion dependency* is a tgd with a single relational atom on each of the left-hand side and right-hand side.

[8] S. Cohen, W. Nutt, and A. Serebrenik. Rewriting aggregate queries using views. In *PODS*, pages 155–166, 1999.

[9] A. Deutsch. *XML Query Reformulation over Mixed and Redundant Storage*. PhD thesis, Univ. Pennsylvania, 2002.

[10] A. Deutsch, A. Nash, and J. Remmel. The chase revisited. In *PODS*, pages 149–158, 2008.

[11] A. Deutsch, L. Popa, and V. Tannen. Query reformulation with constraints. *SIGMOD Record*, 35(1):65–73, 2006.

[12] A. Deutsch and V. Tannen. Reformulation of XML queries and constraints. In *ICDT*, pages 225–241, 2003.

[13] O. M. Duschka and M. R. Genesereth. Answering recursive queries using views. In *PODS*, pages 109–116, 1997.

[14] R. Fagin, P. Kolaitis, R. Miller, and L. Popa. Data exchange: semantics and query answering. *Theoretical Computer Science*, 336(1):89–124, 2005.

[15] H. Garcia-Molina, J. Ullman, and J. Widom. *Database Systems: The Complete Book*. Prentice Hall, 2002.

[16] G. Gou, M. Kormilitsin, and R. Chirkova. Query evaluation using overlapping views: completeness and efficiency. In *SIGMOD Conf.*, pages 37–48, 2006.

[17] A. Halevy. Answering queries using views: A survey. *VLDB Journal*, 10(4):270–294, 2001.

[18] T. Jayram, P. Kolaitis, and E. Vee. The containment problem for *real* conjunctive queries with inequalities. In *PODS*, pages 80–89, 2006.

[19] A. Klug. On conjunctive queries containing inequalities. *Journal of the ACM*, 35(1):146–160, 1988.

[20] A. Levy, A. Mendelzon, Y. Sagiv, and D. Srivastava. Answering queries using views. In *PODS*, 1995.

[21] C. Li. Rewriting queries using views. Encyclopedia of Database Systems, Springer, in print, 2008.

[22] W. Nutt, Y. Sagiv, and S. Shurin. Deciding equivalences among aggregate queries. In *PODS*, pages 214–223, 1998.

[23] J. D. Ullman. Information integration using logical views. *Theoretical Computer Science*, 239(2):189–210, 2000.

APPENDIX

A. THE C&B ALGORITHM OF [10]

In this section of the appendix we give an overview of the Chase and Backchase (C&B) algorithm by Deutsch and colleagues, see [11] for the details. Under set semantics for query evaluation and given a CQ query Q, C&B outputs all equivalent Σ-minimal conjunctive reformulations of Q in presence of the given embedded dependencies Σ, whenever chase of Q under Σ terminates in finite time.

C&B proceeds in two phases. The first phase of C&B, its *chase phase*, does chase of Q using Σ under set semantics, to obtain terminal chase result $(Q)_{\Sigma,S}$. This output of the chase phase is called the *universal plan* U for Q. Note that by construction of U, $Q \equiv_{\Sigma,S} U$.

The second phase of C&B, its *backchase phase*, proceeds as follows:

1. Iterate over all queries U' whose head is $head(U)$ and whose body is not empty and is $body(U)$ with zero or more atoms dropped.
2. Chase each U' using Σ, to obtain terminal chase result $(U')_{\Sigma,S}$.
3. C&B outputs each U' such that for the terminal result $(U')_{\Sigma,S}$ of chasing the candidate reformulation U' under Σ (under set semantics), it holds that $(U')_{\Sigma,S} \equiv_S U$, that is, each U' for which by Theorem 2.2 it holds that $U' \equiv_{\Sigma,S} Q$.

Theorem A.1. **(C&B is sound and complete)** *For an arbitrary instance of the Query-Reformulation Problem with a CQ query Q, set semantics for query evaluation, and a set of embedded dependencies Σ such that chasing Q under Σ terminates in finite time, C&B outputs all Σ-minimal conjunctive reformulations Q' of Q such that $Q' \equiv_{\Sigma,S} Q$.* □

The proof of Theorem A.1 is by construction of C&B.

B. KEYS OF RELATIONS

This section of the appendix provides basic definitions for the standard notion of a key of a relation [15].

B.1 Attributes and Relations

Let $\mathcal{U}$ be a countably infinite set of *attributes.* The *universe* U is a finite subset of $\mathcal{U}$. A *relation schema* R *of arity* k is a subset of U of cardinality k. A *database schema* (or, simply, *schema*) $\mathcal{D}$ over U is a finite set of relation schemas $\{R_1, \ldots, R_t\}$ with union U, of arities $k_1, \ldots, k_t$, respectively.

Each attribute $A \in \mathcal{U}$ has an associated set of *values* $\Delta(A)$, called A's *domain.* The *domain* is the set of values $\Delta = \bigcup_A \Delta(A)$. Let $\mathcal{D}$ be a schema over U, $R \in \mathcal{D}$ a relation schema and X a subset of U. An X*-tuple* t is a mapping from X into Δ, such that each attribute $A \in X$ is mapped to an element of $\Delta(A)$. A (generally bag-valued) *relation* r over R is a finite collection of R-tuples. A *database (instance)* D of $\mathcal{D}$ is a set of relations, with one relation for each relation schema of $\mathcal{D}$.

B.2 Functional Dependencies and Keys

Consider a database schema $\mathcal{D}$ with n-ary relation symbol P such that $n > 1$. A *functional dependency (fd)* on relation P in $\mathcal{D}$ is an egd of the form $p(\bar{X}, Y, \bar{Z}) \wedge p(\bar{X}, Y', \bar{Z}') \rightarrow Y = Y'$, such that predicate p corresponds to relation P. Here, Y and Y' must be in the same position in the respective atoms, meaning the following. Let Y be the ith argument of atom $p(\bar{X}, Y, \bar{Z})$, for some $1 \leq i \leq n$. Then Y' is the ith argument of atom $p(\bar{X}, Y', \bar{Z}')$. Similarly, we require each element of the vector $\bar{X}$ to be in the same position in each of $p(\bar{X}, Y, \bar{Z})$ and $p(\bar{X}, Y', \bar{Z}')$.

Definition B.1. *(***Implied functional dependency***) Let σ be an fd on relation R, and let Σ be a set of fds on R. Then σ* is a functional dependency implied by Σ *if σ holds on all instances of relation R that satisfy Σ.* □

Standard textbooks (see, e.g., [15]) describe algorithms for solving the problem of finding all fds implied by a given set of dependencies on the schema of a relation.

Let $\mathbf{K} = \{A_{i1}, \ldots, A_{ip}\}$ be a nonempty proper subset of the set of attributes of n-ary relation $R(A_1, \ldots, A_n)$, with $n > 1$. That is, $1 \leq p < n$ and $A_{ij} \in \{A_1, \ldots, A_n\}$ for each $j \in \{1, \ldots, p\}$. In the definitions that follow, we will use the following notation: Let $\sigma(\mathbf{K}|A_i)$, for some $i \in \{1, \ldots, n\}$ such that $A_i \notin \mathbf{K}$, denote an fd that equates the values of attribute A_i of R whenever the two r-atoms in the left-hand side of $\sigma(\mathbf{K}|A_i)$ agree on the values of all and only attributes in $\mathbf{K}$. For example, if the schema of R is $R(A, B, C, D)$ and $\mathbf{K} = \{A, C\}$, then $\sigma(A, C|B)$ is defined as

$$\sigma(A, C|B) : r(A, B_1, C, D_1) \wedge r(A, B_2, C, D_2) \rightarrow B_1 = B_2.$$

Definition B.2. *(***Superkey of relation***)* $\mathbf{K}$ *is a* superkey of relation *R if for each attribute A in the set* $\{A_1, \ldots, A_n\} - \mathbf{K}$, *it holds that fd* $\sigma(\mathbf{K}|A)$ *is implied by the set* Σ *of fds on R.* □

The set of all attributes of R is also a superkey of R.

Definition B.3. *(***Key of relation***)* $\mathbf{K}$ *is a* key of relation *R if (1)* $\mathbf{K}$ *is a superkey of R, and (2) for each nonempty proper subset* $\mathbf{K}'$ *of* $\mathbf{K}$, $\mathbf{K}'$ *is not a superkey of R.* □

C. TUPLE IDS FOR RELATIONS

In this section of the appendix we present a solution to the problem of ensuring, under *bag* semantics, that certain base relations are *sets* in all database instances. To this end, we provide here a formal framework for *tuple IDs*, which are unique tuple identifiers commonly used in implementations of real-life database-management systems [15]. Our approach to ensuring that some relations are always set valued is to use functional dependencies (Appendix B) to force certain relations to be set valued, by restricting tuples with the same "contents" (that is, all values with the exception of the tuple ID) to have the same tuple ID.

Assume bag semantics for query evaluation and consider relation symbol R_i in database schema $\mathcal{D}$. (Section B.1 has the relevant definitions.) We follow the approach taken in implementations of real-life database-management systems [15] by incrementing the arity of R_i. As a result, the arity of each relation R_i becomes k_i+1 instead of the original k_i as defined in Section 2.[10]

Let $\mathcal{D}'$ be the schema resulting from such arity modification in $\mathcal{D}$ for each relation R_i. By D' we denote instances of $\mathcal{D}'$. In the schema of R_i in $\mathcal{D}'$, let the last attribute of R_i be the attribute for the tuple ID. The values of all tuple IDs are required to be distinct in all instances of $\mathcal{D}'$, which is formally specified as follows.

Definition C.1. *(***Tuple ID.***) For a relation symbol* R_i *of arity* k_i+1 *in database schema* $\mathcal{D}'$, *let queries* $Q^{R_i}_{tid}$ *and* $Q^{R_i}_{vals}$ *be as follows:*

$$Q^{R_i}_{tid}(X_{k_i+1}) \; :- \; R_i(X_1, \ldots, X_{k_i}, X_{k_i+1}).$$
$$Q^{R_i}_{vals}(X_1, \ldots, X_{k_i}) \; :- \; R_i(X_1, \ldots, X_{k_i}, X_{k_i+1}).$$

Then the (k_i+1)*st attribute of* R_i *in* $\mathcal{D}'$ *is* the tuple ID for R_i *if in all instances* D' *of* $\mathcal{D}'$, *the following relationship holds between the relations* $Q^{R_i}_{tid}(D', B)$ *and* $Q^{R_i}_{vals}(D', B)$:

$$|coreSet(Q^{R_i}_{tid}(D', B))| = |Q^{R_i}_{vals}(D', B)|.$$

Here, coreSet($\mathbf{B}$) *denotes the core-set of bag* $\mathbf{B}$, *and* $|\mathbf{B}|$ *denotes cardinality of* $\mathbf{B}$. □

We now study the relationship between instances D' of $\mathcal{D}'$ and instances D of $\mathcal{D}$. Suppose that for relation R_i of arity k_i+1 in $\mathcal{D}'$, the last attribute of R_i is the tuple ID of R_i. By definition of tuple IDs, for each instance D of $\mathcal{D}$, relation R_i in D can be obtained from some instance D' of $\mathcal{D}'$, by evaluating query $Q^{R_i}_{vals}$ under bag semantics on R_i in D':

$$Q^{R_i}_{vals}(X_1, \ldots, X_{k_i}) \; :- \; R_i(X_1, \ldots, X_{k_i}, X_{k_i+1}).$$

Now suppose that in (the original) schema $\mathcal{D}$, a relation with symbol R_i and arity k_i is required to be set valued in all instances of $\mathcal{D}$. We enforce this requirement by the functional dependency

$$\sigma^{R_i}_{tid}: \; R_i(X_1, \ldots, X_{k_i}, X_{k_i+1}) \wedge R_i(X_1, \ldots, X_{k_i}, Y_{k_i+1}) \rightarrow X_{k_i+1} = Y_{k_i+1}$$

on R_i in schema $\mathcal{D}'$. This functional dependency enforces the same tuple ID for each pair of tuples that agree on the values of all other attributes of R_i. In conjunction with Definition C.1, which ensures uniqueness of each tuple ID within each instance of $\mathcal{D}'$, $\sigma^{R_i}_{tid}$ enforces that the answer to query $Q^{R_i}_{vals}$ (i.e., R_i in schema $\mathcal{D}$) be set valued when computed under bag semantics.

In the context of Example 4.1, in presence of tuple IDs we could formally define dependency σ_6 as an egd:

$$\sigma_6: \; t(X, Y, Z, U) \wedge t(X, Y, Z, W) \rightarrow U = W.$$

Here, the fourth attribute of relation T is the tuple-ID attribute.

D. PROOF OF THEOREM 4.2

This section of the appendix provides a proof of Theorem 4.2. We first supply the details of Example 4.9.

EXAMPLE D.1. *To show that query* Q_3 *of Example 4.1 is not bag equivalent to query* Q_5,

$$Q_3(X) \; :- \; p(X, Y), t(X, Y, W), s(X, Z).$$
$$Q_5(X) \; :- \; p(X, Y), t(X, Y, W), s(X, Z), s(X, Z).$$

we construct a bag-valued database D, with the following relations: $P = \{\{(1,2)\}\}$, $R = \emptyset$, $S = \{\{(1,3),(1,3)\}\}$, $T = \{\{(1,2,5)\}\}$, *and* $U = \emptyset$. *On this database D, the answer to* Q_3 *is* $Q_3(D, B) = \{\{(1),(1)\}\}$, *whereas* $Q_5(D, B) = \{\{(1),(1),(1),(1)\}\}$, *by rules of bag semantics. From the fact that* $Q_3(D, B)$ *and* $Q_5(D, B)$ *are not the same bags, we conclude that bag equivalence* $Q_3 \equiv_B Q_5$ *does not hold.*

At the same time, by Theorem 4.2 it holds that Q_3 *and* Q_5 *are bag equivalent on all databases where relation S is required to be a set.* □

We now prove Theorem 4.2. The If part of the proof is straightforward. For the Only-If part, we argue that the only way for Q_1 and Q_2 to be bag equivalent under the set-enforcing constraints of database schema $\mathcal{D}$ is for Q_1 and Q_2 to satisfy the conditions of Lemma D.1. The proof of Lemma D.1 completes the proof of Theorem 4.2, by showing by contrapositive that bag equivalence of Q_1 and Q_2 under the set-enforcing constraints of database schema $\mathcal{D}$ has to entail isomorphism of the queries Q_1' and Q_2' defined in the statement of Theorem 4.2.

Proof. (Theorem 4.2)

If. Let database schema $\mathcal{D}$ have a relation symbol P, such that the relation for P is set valued in all (bag-valued) instances D over $\mathcal{D}$. (Appendix C provides an approach to enforcing this set-valuedness constraint using functional dependencies that involve tuple IDs.)

[10]We emulate the standard implementation practice that tuple IDs be invisible to the users of the database system; that is, in our approach the user assumes that the arity of each relation R_i is still k_i.

Consider an arbitrary CQ query Q_1 that has a subgoal with predicate p corresponding to relation P; w.l.o.g. let the subgoal be $p(\bar{W})$. Let Q_2 be a CQ query obtained by adding to the body of Q_1 a duplicate of $p(\bar{W})$.

We argue that for Q_1 and Q_2 as described above, it holds that $Q_1 \equiv_B Q_2$ under the set-enforcing dependencies of the schema $\mathcal{D}$. (The claim of the If direction of the theorem is immediate from this observation.) Indeed, consider an arbitrary instance D of database schema $\mathcal{D}$, such that D satisfies the set-enforcing dependencies of the schema $\mathcal{D}$. From the definition of bag semantics for query evaluation it follows that each assignment satisfying the body of Q_1 w.r.t. D is also a satisfying assignment for the body of Q_2 w.r.t. D, and vice versa. Further, each such satisfying assignment γ maps $p(\bar{W})$, in the body of Q_1, into a *single* tuple t in relation P in D, and similarly γ maps *both copies* of $p(\bar{W})$, in the body of Q_2, into the same single tuple t, due to relation P being set valued in the database D. It follows that each such satisfying assignment γ contributes to each of $Q_1(D,B)$ and $Q_2(D,B)$ *the same* number of tuples under bag semantics for query evaluation. The claim of the If direction of Theorem 4.2 is immediate from the above observation.

Only-If. The proof is by contrapositive. For two CQ queries Q_1 and Q_2, let $Q_1 \equiv_B Q_2$ hold in the absence of all dependencies other than the set-enforcing dependencies of the schema $\mathcal{D}$. Consider queries Q_1' and Q_2' defined in the statement of Theorem 4.2. We assume that Q_1' and Q_2' are not isomorphic, and obtain from this assumption that Q_1 and Q_2 are not bag equivalent on at least one database that satisfies the set-enforcing dependencies of schema $\mathcal{D}$, in contradiction with what we are given.

W.l.o.g., let s be a subgoal of query Q_1' such that either Q_2' has no subgoals with the predicate of s, or Q_2' has fewer (but still a positive number of) subgoals with the predicate of s than Q_1' does. Consider first the case where Q_2' has no subgoals with the predicate of s; it follows from the construction of queries Q_1' and Q_2' that Q_2 does not have subgoals with the predicate of s either, whereas Q_1 has at least one occurrence of subgoal with the predicate of s. Observe that in this case, *set* equivalence between Q_1 and Q_2 does not hold by the results of [2]. From the result of [4] (see Proposition 2.1 in this current paper) that bag equivalence implies set equivalence, it follows immediately that bag equivalence of Q_1 and Q_2 cannot hold either, *in presence* of the set-enforcing dependencies in the schema $\mathcal{D}$. (This follows from the fact that $Q_2 \not\sqsubseteq_S Q_1$ implies existence of a *set-valued* database on which Q_2 under *set* semantics produces a tuple t, such that t is *not* in the *set*-semantics answer to Q_1 on the same database.) Thus, we have arrived at a contradiction with our assumption that $Q_1 \equiv_B Q_2$ on all databases satisfying the set-enforcing dependencies of the schema $\mathcal{D}$.

We now consider the remaining case concerning the number in Q_2' of subgoals with the predicate of s, that is the case where Q_2' has fewer (but still a positive number of) subgoals with the predicate of s than Q_1' does. Suppose first that there is no *bag-set* equivalence between Q_1 and Q_2. That is, by Theorem 2.1 we assume that the canonical representations of Q_1' and of Q_2' (which are the same as the canonical representations of Q_1 and of Q_2, respectively) are not isomorphic. Then similarly to the previous case considered in this proof, from Proposition 2.1 we obtain immediately the contradiction to $Q_1 \equiv_B Q_2$ under the set-enforcing dependencies of schema $\mathcal{D}$. (Similarly to the case above, $Q_1 \equiv_{BS} Q_2$ would have to be violated on a *set*-valued database, therefore the set-enforcing dependencies of the schema $\mathcal{D}$ would be satisfied in that $Q_1 \equiv_B Q_2$ would be violated on the *same* database.)

Thus, for the rest of this proof we assume that (1) $Q_1 \equiv_{BS} Q_2$, and (2) $Q_1' \equiv_{BS} Q_2'$ (from $Q_1 \equiv_{BS} Q_2$ and by construction of Q_1' and Q_2'). That is, for both pairs of queries the canonical representations are isomorphic. Under these restrictions, the only way Q_1' and Q_2' can be nonisomorphic is the case where Q_1' (w.l.o.g.) has more subgoals (than Q_2') whose predicate corresponds to a relation, say R, that is *not* required to be a set in all instances of schema $\mathcal{D}$. (Indeed, if Q_1 and Q_2 have this number-of-subgoals discrepancy for a predicate whose relation *is* required to be a set in all instances of $\mathcal{D}$, then Q_1' and Q_2' must have the same number of such subgoals by $Q_1 \equiv_{BS} Q_2$ and by construction of Q_1' and Q_2'.) Note that in this case, relation symbol R must belong to $\mathcal{D} - \{P_1, \ldots, P_k\}$ ("$-$" is set difference), and thus the subset relationship $\{P_1, \ldots, P_k\} \subseteq \mathcal{D}$ is proper in this case, that is $\{P_1, \ldots, P_k\} \subset \mathcal{D}$. Recall that $\{P_1, \ldots, P_k\}$ is the maximal subset of $\mathcal{D}$ such that all symbols in $\{P_1, \ldots, P_k\}$ correspond to relations required to be set valued in all instances of $\mathcal{D}$.

We finish the proof of Theorem 4.2 by proving Lemma D.1, which constructs a database D satisfying the set-enforcing dependencies of schema $\mathcal{D}$. By construction, database D is a counterexample to $Q_1 \equiv_B Q_2$ (on databases satisfying the set-enforcing dependencies of schema $\mathcal{D}$), whenever Q_1' has more subgoals (than Q_2') whose predicate corresponds to a relation that is not required to be a set in all instances of schema $\mathcal{D}$. □

LEMMA D.1. *Let $\mathcal{D}$, $\{P_1, \ldots, P_k\} \subset \mathcal{D}$, Q_1, Q_2, Q_1', and Q_2' be as specified in Theorem 4.2, and let $Q_1 \equiv_{BS} Q_2$. Let R be a relation symbol in the set $\mathcal{D} - \{P_1, \ldots, P_k\}$; that is, relation R is not required to be a set in all instances of $\mathcal{D}$. Suppose that Q_1' has strictly more subgoals whose predicate corresponds to R than Q_2' does. Then there exists an instance D of $\mathcal{D}$ such that all of relations $P_1, \ldots, P_k$ are set valued in D, and such that $Q_1(D,B)$ is not the same bag as $Q_2(D,B)$.* □

By the above characterization, database D is a counterexample to queries Q_1 and Q_2 being bag equivalent on all instances of $\mathcal{D}$ that satisfy the set-enforcing restrictions of schema $\mathcal{D}$.

The intuition for the proof of Lemma D.1 is as follows. Let query Q_1 have $n_1 > 1$ subgoals whose predicate corresponds to relation R, such that R is not required to be set valued in instances of schema $\mathcal{D}$. (Part of the proof is to show that by the properties of this relation symbol R and by construction of Q_1' from Q_1, it holds that Q_1 and Q_1' have exactly the same number of subgoals whose predicate corresponds to R. We make the same observation about Q_2 and Q_2'.) Further, let query Q_2 have a positive number (by proof of Theorem 4.2) $n_2 < n_1$ of subgoals whose predicate corresponds to R. We build a database D on which Q_1 produces at least $m^{(n_1)}$ copies of some (distinct) tuple t^*, with the positive integer value of m to be determined. We then "let"

Q_2 have as many satisfying assignments for the body of Q_2 w.r.t. this database D as possible. That is, we assume the best case for Q_2 of producing as many tuples on database D as possible. We then show that if the value of m is chosen in a certain way, then the number $m^{(n_1)}$ of copies of tuple t^* in the bag $Q_1(D,B)$ is greater than the maximal (i.e., best-case) number N of *all* tuples (counting all duplicate tuples as separate tuples) that can be contributed by Q_2 to the bag $Q_2(D,B)$. The reason that we can make such a choice of the value of m is that this maximal number N grows asymptotically as $m^{(n_2)}$, with $0 < n_2 < n_1$, whereas the number of copies of tuple t^* in the bag $Q_1(D,B)$ is $m^{(n_1)}$.

PROOF. (Lemma D.1) Let n_1 be the number of subgoals in Q_1' whose predicate corresponds to R, and let n_2 be the number of subgoals in Q_2' whose predicate corresponds to R; $n_2 > 0$ by $Q_1 \equiv_{BS} Q_2$. By our assumption, $n_1 \geq n_2 + 1$. By construction of Q_1', Q_1 has the same number n_1 of subgoals whose predicate corresponds to R as Q_1' does; we make the same observation about the relationship between the number n_2 of subgoals in Q_2 whose predicate corresponds to R and the (same) number n_2 of subgoals of the same type in Q_2'. (See proof of Theorem 4.2 for the details of the argument.)

Let D' be the (set-valued by definition, see Section 2.1) canonical database for the canonical representation of Q_1'. (From the proof of Theorem 4.2, we have that Q_1, Q_1', Q_2, and Q_2' all have the same canonical representation.) We construct from D' our counterexample database D as follows.

1. For each relation symbol S in $\mathcal{D} - \{R\}$, the relation S in D is the same as the relation S in D'. By construction of D', all the relations in $\{P_1, \ldots, P_k\}$ are *set valued* in database D. Thus, database D satisfies the set-enforcing restrictions of the schema $\mathcal{D}$.

2. We build relation R in D by "putting together" $m > 0$ copies of relation R in D', with the value of m to be determined shortly. That is, for each tuple t such that t is in the *set-valued* relation R in D', relation R in D has m copies of tuple t; further, R in D has no other tuples.

By definition of bag semantics for query evaluation, see Section 2.2, the bag $Q_1(D,B)$ has at least $m^{(n_1)}$ copies of some individual tuple. Indeed, consider the assignment mapping γ from Q_1 to D such that γ was used to *generate* the canonical database D' of the canonical representation of Q_1. (See Section 2.1 for the description of the process of construction of a canonical database for a CQ query.) Observe that γ is a satisfying assignment for the body of Q_1 w.r.t. database D. The assignment γ maps each of the R-subgoals of Q_1 to at least m tuples of R, by construction of relation R in D, and γ maps each non-R subgoal (if any) of Q_1 to exactly one tuple. Thus, for the tuple $t^* = \gamma(\bar{X}) \in Q_1(D,B)$, where $Q_1(\bar{X})$ is the head of the query Q_1, the multiplicity of t^* in $Q_1(D,B)$ is at least $m^{(n_1)}$. (The "at least" part comes from the possibility that extra copies of the tuple t^* could be contributed to the bag $Q_1(D,B)$ by one or more satisfying assignments γ' for the body of Q_1 w.r.t. database D, such that each such γ' is *not* identical to the assignment γ.)

At the same time, we show that the total size of the bag $Q_2(D,B)$ cannot exceed

$$n_1^{(2n_2)} \times n_4^{(n_3-n_2)} \times m^{(n_2)} \tag{4}$$

tuples, in case the total number n_3 of subgoals of query Q_2 is greater than n_2; n_4 is the number of subgoals of Q_1 whose (subgoals') predicate does not correspond to relation symbol R. (By $Q_1 \equiv_{BS} Q_2$ we have that $n_4 > 0$ whenever $n_3 > n_2$.) In this case, we set the value m^* of m to

$$m^* \ := \ 1 + n_1^{(2n_2)} \times n_4^{(n_3-n_2)}. \tag{5}$$

It follows that

$$(m^*)^{(n_1-n_2)} > n_1^{(2n_2)} \times n_4^{(n_3-n_2)}. \tag{6}$$

That is (recall that $0 < n_2 < n_1$),

$$(m^*)^{(n_1)} > n_1^{(2n_2)} \times n_4^{(n_3-n_2)} \times (m^*)^{(n_2)}. \tag{7}$$

We conclude that on the database D where the value of m is fixed at m^*, the number of copies of tuple t^* in the bag $Q_1(D,B)$ exceeds the number of *all* tuples in the bag $Q_2(D,B)$. Therefore, the bag $Q_1(D,B)$ is not the same bag as $Q_2(D,B)$.

(In case the total number n_3 of subgoals of query Q_2 is equal to n_2, we show that the bag $Q_2(D,B)$ cannot have more than

$$n_1^{(2n_2)} \times m^{(n_2)} \tag{8}$$

tuples. In this case, we set the value m^* of m to

$$m \ := \ 1 + n_1^{(2n_2)}. \tag{9}$$

It follows that at this value m^* of m, we have that

$$(m^*)^{(n_1-n_2)} > n_1^{(2n_2)}. \tag{10}$$

That is (recall that $0 < n_2 < n_1$),

$$(m^*)^{(n_1)} > n_1^{(2n_2)} \times (m^*)^{(n_2)}. \tag{11}$$

We conclude that on the database D where the value of m is fixed at m^*, the number of copies of tuple t^* in the bag $Q_1(D,B)$ exceeds the number of *all* tuples in the bag $Q_2(D,B)$. Therefore, the bag $Q_1(D,B)$ is not the same bag as $Q_2(D,B)$. The proof of this case is straightforward from the proof, see below, of Equation 4 for the case where the total number n_3 of subgoals of query Q_2 is greater than n_2.)

We now explain why the bag $Q_2(D,B)$ cannot be of greater cardinality than the number of tuples specified in Equation 4, in the case where the total number n_3 of subgoals of query Q_2 is greater than n_2. The idea of the proof is to "let" Q_2 have as many satisfying assignments for the body of Q_2 w.r.t. database D as possible. That is, we assume the best case for Q_2 of producing as many tuples on database D as possible. We take the following specific steps in building the upper bound:

1. We assume the best case for Q_2 of the number of satisfying assignments, w.r.t. database D, for the $n_3 - n_2$ subgoals of Q_2 whose (subgoals') predicates do not correspond to R. The maximal number of such assignments cannot exceed

$$n_4^{(n_3-n_2)}. \tag{12}$$

That is, the best case for Q_2 is to assume that all of the $n_3 - n_2$ subgoals of Q_2 have *the same* predicate, say predicate s corresponding to the relation symbol S, where S may or may not be one of the relation symbols $P_1, \ldots, P_k$ specified in the formulation of this Lemma. We also assume that the $n_4 > 0$ non-R subgoals of Q_1 also have the same predicate s. Database D has at most n_4 tuples in relation S (by construction of canonical databases). We assume the best case for Q_2 that each of the $n_3 - n_2$ subgoals of Q_2 can map independently into each of the (at most) n_4 tuples, hence the formula of Equation 12.

2. For each of the above $n_4^{(n_3-n_2)}$ assignments, Q_2 may have at most
$$n_1^{(n_2)} \tag{13}$$
satisfying assignments, w.r.t. database D, for all the n_2 subgoals of Q_2 whose predicate corresponds to the relation symbol R. The computations are similar to those that we used in explaining Equation 12.

3. For each of the $n_1^{(n_2)}$ satisfying assignments, w.r.t. database D, for all the n_2 subgoals of Q_2 whose predicate corresponds to the relation symbol R, Q_2 can produce on database D at most
$$(n_1 \times m)^{(n_2)} \tag{14}$$
tuples. We obtain the formula of Equation 14 by assuming that the evaluation of Q_2 admits a Cartesian product of n_2 copies of the relation R, where relation R has at most $n_1 \times m$ tuples on D.

4. We combine Equations 12, 13, and 14, to obtain that the total number of satisfying assignments for the body of Q_2 w.r.t. database D cannot exceed
$$n_1^{(n_2)} \times n_4^{(n_3-n_2)} \tag{15}$$
(satisfying assignments); and that, further, for each one of these assignments Q_2 produces on database D at most
$$(n_1 \times m)^{(n_2)} \tag{16}$$
tuples (where each duplicate is counted separately) in the bag $Q_2(D, B)$. (Recall that all relations except R are set valued in database D.) We conclude that the total number of tuples (including duplicates) that query Q_2 produces on database D is at most
$$(n_1)^{2(n_2)} \times n_4^{(n_3-n_2)} \times m^{(n_2)} \tag{17}$$
tuples. Equation 17 gives us an upper bound on the size of the bag $Q_2(D, B)$. Q.E.D.

□

Consider an illustration to the proof of Lemma D.1.

EXAMPLE D.2. *Let CQ queries Q_7 and Q_8 be defined as*

$Q_7(X)\ :-\ p(X,Y), r(X), r(X).$
$Q_8(X)\ :-\ p(X,Y), r(X).$

in the setting of Example 4.1. To illustrate the proof of Lemma D.1, we construct a counterexample database to the claim that Q_7 and Q_8 are bag equivalent on all databases that satisfy just the set-enforcing dependencies of Example 4.1. We use the fact that query Q_7 has two copies of subgoal $r(X)$, whereas Q_8 has just one copy of that subgoal. (Recall that relation R is not required to be a set on all instances of the database schema $\mathcal{D}$ of Example 4.1.)

Queries Q_7 and Q_8, as well as the database schema $\mathcal{D}$ of Example 4.1 together with its set-enforcing constraints, satisfy all the conditions of Lemma D.1. Observe that query Q'_7 (defined in the statement of Theorem 4.2) is isomorphic to Q_7, because relation R is not required to be a set. Similarly, query Q'_8 is isomorphic to Q_8. Further, the canonical representation of each of Q_7, Q_8, Q'_7, and Q'_8 is isomorphic to query Q_8.

Consider query Q'_8 and its canonical database D', with $P = \{\{(1,2)\}\}$ and $R = \{\{(1)\}\}$. From D', we construct a bag-valued database D, with relations $P = \{\{(1,2)\}\}$ (same as P in D') and with $m > 0$ copies of tuple (1) in relation R. That is, $R = \{\{(1), \ldots, (1)\}\}$, with cardinality m of bag R in D. Let relations S, T, U be empty sets in D. Then D satisfies all the set-enforcing dependencies of Example 4.1.

Now using the notation of the proof of Lemma D.1, we have $n_1 = 2$. Here, n_1 is the number of subgoals of Q'_7 – and thus also of Q_7 – whose predicate corresponds to R. At the same time, $n_2 = 1 < n_1$, where n_2 is the number of subgoals of Q'_8 – and thus also of Q_8 – whose predicate corresponds to R. Further, the total number n_3 of subgoals of Q_8 is $n_3 = 2$, and the number n_4 of non-R subgoals of Q_7 is $n_4 = 1$.

It is easy to verify that the bag $Q_7(D,B)$ has $m^{(n_1)} = m^2$ copies of tuple (1). At the same time, by the argument justifying Equation 4 in the proof of Lemma D.1, the total number of tuples (where each duplicate is counted separately) in the bag $Q_8(D,B)$ cannot exceed
$$n_1^{(2n_2)} \times n_4^{(n_3-n_2)} \times m^{(n_2)} = 2^2 \times 1^{(2-1)} \times m^1 = 4m$$
tuples. It is easy to see that for any value m^* of m such that $m^* > 4$, the number of copies of tuple (1) in the bag $Q_7(D,B)$ is always going to be greater than the cardinality of the bag $Q_8(D,B)$.

In fact, the upper bound of Equation 4 is not tight for this example, as can be observed from the facts that

- *the total number of copies of tuple (1) in bag $Q_8(D,B)$ is m, and*
- *the core-set of the bag $Q_8(D,B)$ has no tuples other than (1); therefore, the cardinality of the bag $Q_8(D,B)$ is m as well.*

□

E. PROOFS OF THE THEOREMS ON SOUND CHASE STEPS

We provide here representative parts of proofs for Theorems 4.1 and 4.3. The idea of the complete proofs is to show, for an arbitrary embedded dependency, one of the following two things:

(1) Either using the dependency results in sound chase steps, under the appropriate semantics, for all CQ queries, in case the format of the dependency is described in the applicable theorem (i.e., either Theorem 4.1 or Theorem 4.3). Please see Proposition E.1 in Section E.2 for an example of such a claim.

(2) Or using the dependency results in unsound chase, in case the format of the dependency is not described in the theorem for the respective query-evaluation semantics (i.e., either Theorem 4.1 or Theorem 4.3). Please

see Propositions E.2 and E.3 in Section E.2 for examples of such claims.

All the remaining proofs for Theorems 4.1 and 4.3 are analogous to the proofs of Propositions E.1 through E.3.

E.1 Bag Projection

This subsection of the appendix defines bag projection. We use the definition in the proof of Proposition E.1 in Section E.2.

Given positive integers m, k and $i(1), \ldots, i(k)$, such that for each $j \in \{1, \ldots, k\}$ it holds that $1 \leq i(j) \leq m$. Then for an m-tuple $t = (a_1, \ldots, a_m)$, we say that a k-tuple $t' = (a_{i(1)}, \ldots, a_{i(k)})$ is a *projection of t on attributes in positions* $i(1), \ldots, i(k)$, denoted
$t' = t[i(1), \ldots, i(k)]$.
Further, for the m, k and $i(1), \ldots, i(k)$ as above and for an m-ary relation P, a bag of tuples B is a *bag projection of P on attributes in positions* $i(1), \ldots, i(k)$, denoted $B = \pi^{bag}_{i(1),\ldots,i(k)}(P)$, if each tuple $t \in P$ contributes to B a separate tuple $t' = t[i(1), \ldots, i(k)]$, and if B has no other tuples. B can be interpreted as the answer $Q(D, B)$ on database $\{P\}$ to query
$Q(X_{i(1)}, \ldots, X_{i(k)}) \ :- \ p(X_1, \ldots, X_m)$,
where the predicate p corresponds to relation P.

E.2 The Proofs

Proposition E.1. *Given a CQ query Q and a set of embedded dependencies Σ. Under bag semantics for query evaluation, a chase step $Q \Rightarrow^{\sigma}_{B} Q'$ using tgd $\sigma \in \Sigma$ is sound if $Q \Rightarrow^{\sigma}_{B} Q'$ is (tgd) key-based, and for each subgoal $s(p_{ij})$ that the chase step adds to Q, relation P_{ij} is set valued on all databases satisfying Σ.* □

Proof. Let σ be of the form

$\sigma : \phi(\bar{X}, \bar{Y}) \rightarrow \exists \bar{Z} \ p_1(\bar{Y}_1, \bar{Z}_1) \wedge \ldots \wedge p_n(\bar{Y}_n, \bar{Z}_n)$,

with $n > 0$. Here, the set of variables in each $\bar{Y}_i$, $i \in \{1, \ldots, n\}$, is the maximal subset, in the set of variables in $\bar{Y}_i \bigcup \bar{Z}_i$, of the set of variables in $\bar{Y}$. (We abuse the notation by treating $\bar{Y}_i \bigcup \bar{Z}_i$ as a set of variables and constants.) We show that the chase step $Q \Rightarrow^{\sigma}_{B} Q'$ using σ is sound whenever for all $i \in \{1, \ldots, n\}$ such that $p_i(\bar{Y}_i, \bar{Z}_i)$ corresponds to a subgoal in Q' that is not a subgoal of Q, it holds that (1) $\bar{Y}_i$ is a superset of the key of relation symbol P_i in $\mathcal{D}$, and (2) P_i is set valued in all databases with schema $\mathcal{D}$.

By our assumption that σ is applicable to Q, (1) there exists a mapping μ from a (not necessarily proper) superset ξ of ϕ to a subset of subgoals of Q. By the same assumption, (2) there does not exist a mapping μ' such that μ' is an extension of μ and such that $\mu'(\psi)$ is also a subset of subgoals of Q. Here, ψ is the right-hand side of the tgd σ.

Consider a mapping ν from ϕ to the body of Q, such that ν agrees with μ on all the variables in ξ (note that all of $\bar{Y}$ are in ξ), and such that ν maps the subset of variables $\bar{Z}$ in $\psi - \xi$ (here, "$\psi - \xi$" is read as set difference between sets of conjuncts ψ and ξ) into distinct fresh variables. By definition of chase step for tgds, $\nu(\psi)$ adds at least one subgoal to Q, which results in query Q'. Let one such new subgoal S be the result of applying ν to atom $p_i(\bar{Y}_i, \bar{Z}_i)$ in the ψ part of σ, for some $i \in \{1, \ldots, n\}$.

Consider an arbitrary database D with schema $\mathcal{D}$, such that D satisfies the dependencies Σ. To finalize our proof, it remains to show that on D, the following two relations are the same as bags: $Q(D, B)$ and $Q''(D, B)$, where Q'' results from adding the subgoal S to the body of Q. Here, each of $Q(D, B)$ and $Q''(D, B)$ is to be computed under bag semantics for query evaluation.

Let $bQ(D, B)$ be the relation, on D, for the body of $Q(D, B)$, and let $bQ''(D, B)$ be the relation, on D, for the body of $Q''(D, B)$. Note that if $bQ(D, B)$ and $bQ''(D, B)$ are the same bags modulo the columns of $bQ(D, B)$, then $Q(D, B)$ and $Q''(D, B)$ are the same bags as well. (Recall that the heads of Q and Q'' are the same by definition of Q''.) When we say "$bQ(D, B)$ and $bQ''(D, B)$ are the same bags modulo the columns of $bQ(D, B)$", the meaning is as follows: If we do bag projection on $bQ''(D, B)$ on just the columns of $bQ(D, B)$, then we will obtain precisely $bQ(D, B)$. (Please see Appendix E.1 for the definition of bag projection.)

We now show that $bQ(D, B)$ and $bQ''(D, B)$ are the same bags modulo the columns of $bQ(D, B)$, which finalizes our proof. The case where $bQ(D, B)$ is empty is trivial, thus we assume for the remainder of the proof that $bQ(D, B)$ is not an empty bag. Consider an assignment mapping λ that was used to obtain a tuple t in bag $bQ(D, B)$. By definition of (tgd) key-based chase step for σ, there is exactly one way (up to duplicates of stored tuples) to extend λ, to obtain a (distinct) tuple $t' \in P_i$, such that t' "matches" t according to the join conditions between the body of Q and the new subgoal S in Q''.[11] Further, from the fact that the relation P_i is a set on D, we obtain that t' is a unique tuple (i.e., it has no duplicates in P_i) that "matches" t in the above sense. As a result, each single tuple in $bQ(D, B)$ corresponds, for the purposes of computing $bQ''(D, B)$ from $bQ(D, B)$, to exactly one tuple in P_i.

Observe that the above procedure for computing $bQ''(D, B)$ from $bQ(D, B)$ corresponds to a valid plan for computing $bQ''(D, B)$ from only the stored relations in D. (This plan is a left-linear plan, such that P_i is the right input of the top join-operator node in the tree. For the basics on query-evaluation plans, please see [15].) We conclude that $bQ(D, B)$ and $bQ''(D, B)$ are the same bags modulo the columns of $bQ(D, B)$. □

Proposition E.2. *Given a CQ query Q and a set of embedded dependencies Σ. Consider a key-based chase step $Q \Rightarrow^{\sigma}_{B} Q'$ using tgd $\sigma \in \Sigma$,*

$\sigma : \phi(\bar{X}, \bar{Y}) \rightarrow \exists \bar{Z} \ \psi(\bar{Y}, \bar{Z})$.

Suppose that at least one relation P_i used in ψ is not set valued. Further, suppose that in the chase step $Q \Rightarrow^{\sigma}_{B} Q'$ using σ, Q' is obtained by adding to the body of Q a new P_i-subgoal $s(P_i)$ (possibly alongside other subgoals).[12] *Then under bag semantics for query evalua-*

[11]That is, the extension of λ is a satisfying assignment for the body of Q'' w.r.t database D. In this and other proofs, we can use "procedural" evaluation of queries under each of bag and bag-set semantics. The correctness of this usage stems from the fact that our definitions for query evaluation under bag and bag-set semantics, see Section 2.2, are consistent with the operational semantics of evaluating CQ queries in the SQL standard, as shown in [4].

[12]I.e., in the chase step $Q \Rightarrow^{\sigma}_{B} Q'$, applying σ to Q may generate other new subgoals besides the P_i-subgoal.

tion, the chase step $Q \Rightarrow^{\sigma}_{B} Q'$ using σ is not sound. $\square$

PROOF. Let σ be of the form

$$\sigma : \phi(\bar{X}, \bar{Y}) \rightarrow \exists \bar{Z}\ p_1(\bar{Y}_1, \bar{Z}_1) \wedge \ldots \wedge p_n(\bar{Y}_n, \bar{Z}_n),$$

with $n > 0$. Here, for each $j \in \{1, \ldots, n\}$, $\bar{Y}_j$ is the maximal subset of $\bar{Y}$ in the set $\bar{Y}_j \bigcup \bar{Z}_j$, please see proof of Proposition E.1 for the notation. In addition, the relation P_i for $p_i(\bar{Y}_i, \bar{Z}_i)$ is not a set-valued relation for at least one $i \in \{1, \ldots, n\}$. Given this assumption on P_i, the proof of the claim of Proposition E.2 is by providing a bag-valued database D, such that D satisfies Σ and such that $Q(D, B)$ and $Q'(D, B)$ are not the same bags.

We build the database D as follows. Let D' be the canonical database for query Q'. We obtain D by adding to D' a single duplicate of the tuple for the subgoal $s(P_i)$ of Q'. We now follow the reasoning in the proof of Proposition E.1, to observe that the bag $Q'(D, B)$ has *at least one more tuple than* the bag $Q(D, B)$, due to the fact that the two identical tuples of relation P_i add to $Q'(D, B)$ an extra copy of at least one tuple in $Q(D, B)$. This observation concludes the proof. $\square$

We now provide an illustration that shows the main points of the proof of Proposition E.2.

EXAMPLE E.1. *Consider a set $\Sigma = \{\sigma_1, \sigma_2\}$ of embedded dependencies, where*

$$\sigma_1 : p(X, Y) \wedge p(X, Z) \rightarrow Y = Z.$$
$$\sigma_2 : r(X, Y) \rightarrow p(X, Y).$$

Observe that chase steps using σ_2 are (tgd) key-based in presence of the egd σ_1, and that Σ does not include dependencies that would restrict the relation P, in the right-hand side of σ_2, to be set valued.

Consider a CQ query Q defined as

$$Q(A)\ :-\ r(A, B).$$

Applying σ_2 to the query Q, in chase step $Q \Rightarrow^{\sigma_2}_{B} Q'$, results in query Q' defined as

$$Q'(A)\ :-\ r(A, B), p(A, B).$$

We now illustrate the construction of the database D in the proof of Proposition E.2. First, the canonical database D' of Q' has relations $R = \{\!\{(a, b)\}\!\}$ and $P = \{\!\{(a, b)\}\!\}$. D is constructed from D' by adding to relation P a duplicate of the tuple (a, b), that is D has relations $R = \{\!\{(a, b)\}\!\}$ and $P = \{\!\{(a, b), (a, b)\}\!\}$. Note that database D is bag valued and satisfies all the dependencies in Σ.

Now by the bag semantics for query evaluation, $Q(D, B) = \{\!\{(a)\}\!\}$, while $Q'(D, B) = \{\!\{(a), (a)\}\!\}$. Thus, database D is a counterexample to $Q \equiv_{\Sigma,B} Q'$, which proves that the chase step $Q \Rightarrow^{\sigma_2}_{B} Q'$ using σ_2 is not sound. $\square$

PROPOSITION E.3. *Given a CQ query Q and a set of embedded dependencies Σ. Let $\sigma \in \Sigma$ be a tgd,*

$$\sigma : \phi(\bar{X}, \bar{Y}) \rightarrow \exists \bar{Z}\ p_1(\bar{Y}_1, \bar{Z}_1) \wedge \ldots \wedge p_n(\bar{Y}_n, \bar{Z}_n),$$

with $n > 0$.[13] *Consider a chase step $Q \Rightarrow^{\sigma}_{BS} Q'$ using σ, such that Q' is obtained by adding to the body of Q a new P_i-subgoal $s(P_i)$ (possibly alongside other subgoals), where $s(P_i)$ corresponds to conjunct $p_i(\bar{Y}_i, \bar{Z}_i)$ in the consequent ψ of σ. Suppose that $\bar{Y}_i$ is not a superkey of P_i. Then under bag-set semantics, the chase step $Q \Rightarrow^{\sigma}_{BS} Q'$ using σ is not sound.* $\square$

[13]For the notation, please see proof of Proposition E.1.

(Proposition E.3 is formulated for the case of bag-set semantics, which allows us to show the flavor of the proofs that are required to establish Theorem 4.3.)

PROOF. (Proposition E.3) Given the assumption that for the conjunct $p_i(\bar{Y}_i, \bar{Z}_i)$ used in ψ, it holds that $\bar{Y}_i$ is not a superkey of P_i, the proof of the claim of Proposition E.3 is by providing a *set*-valued database D, such that D satisfies Σ and such that $Q(D, BS)$ and $Q'(D, BS)$ are not the same bags.

Fix i such that $\bar{Y}_i$ in $p_i(\bar{Y}_i, \bar{Z}_i)$ is not a superkey of P_i and such that $p_i(\bar{Y}_i, \bar{Z}_i)$ corresponds to a subgoal in Q' that (subgoal) is not in Q, in chase step $Q \Rightarrow^{\sigma}_{BS} Q'$. We begin the construction of the database D by building the canonical database D' for query Q'. We obtain D by adding to D' a single extra *(nonduplicate)* tuple for the subgoal $s(P_i)$ of Q', as follows.

Without loss of generality, let $p_i(\bar{Y}_i, \bar{Z}_i)$ be of the form $p_i(\bar{Y}_i, \bar{Z}'_i, \bar{Z}''_i)$, where $\bar{Z}'_i$ is not empty, $\bar{Y}_i \bigcup \bar{Z}'_i$ is a superkey of P_i, and no proper subset of $\bar{Y}_i \bigcup \bar{Z}'_i$ is a superkey of P_i. Now suppose ν was the mapping used to generate $s(P_i)$ from $p_i(\bar{Y}_i, \bar{Z}'_i, \bar{Z}''_i)$ in the chase step $Q \Rightarrow^{\sigma}_{BS} Q'$. (See proof of Proposition E.1 for the details on ν.) Then $s(P_i)$ is of the form $p_i(\bar{A}, \bar{C}, \bar{E})$, where $\bar{A}$ ($\bar{C}$, $\bar{E}$, respectively) is the image of $\bar{Y}_i$ (of $\bar{Z}'_i$, of $\bar{Z}''_i$, respectively) under ν. By construction of the canonical database D' of Q', the tuple for $s(P_i)$ in relation P_i in D' is $(\bar{a}, \bar{c}, \bar{e})$. We construct the database D from D' by adding to P_i of D' a tuple $(\bar{a}, \bar{c}', \bar{e})$, such that at least one constant in $\bar{c}'$ is not equal to the same-position constant in $\bar{c}$. By construction, database D is set valued and satisfies the dependencies Σ. (Recall that no proper subset of $\bar{Y}_i \bigcup \bar{Z}'_i$ in $p_i(\bar{Y}_i, \bar{Z}'_i, \bar{Z}''_i)$ is a superkey of P_i.)

We now follow the reasoning in the proof of Proposition E.1, to observe that the bag $Q'(D, BS)$ has *at least one more tuple than* the bag $Q(D, BS)$. The reason is, tuples $(\bar{a}, \bar{c}, \bar{e})$ and $(\bar{a}, \bar{c}', \bar{e})$ in relation P_i add to $Q'(D, BS)$ an extra copy of at least one tuple in $Q(D, BS)$. This observation concludes the proof. $\square$

We now provide an illustration that shows the main points of the proof of Proposition E.3.

EXAMPLE E.2. *Consider a set $\Sigma = \{\sigma\}$ of embedded dependencies, where*

$$\sigma : r(X, Y) \rightarrow p(X, Z).$$

Given a CQ query Q,

$$Q(A)\ :-\ r(A, B).$$

applying σ to the query Q results in query Q',

$$Q'(A)\ :-\ r(A, B), p(A, C).$$

Observe that chase step $Q \Rightarrow^{\sigma}_{BS} Q'$ using σ is not key-based, as the set of all attributes of P is the only key of P.

We now illustrate the construction of the database D in the proof of Proposition E.3. First, the canonical *database D' of Q' has relations $R = \{(a,b)\}$ and $P = \{(a,c)\}$. D is constructed from D' by adding to relation P a new tuple (a,d), that is D has relations $R = \{(a,b)\}$ and $P = \{(a,c),(a,d)\}$. Note that database D is set valued and satisfies the dependency σ.*

Now by the bag-set semantics for query evaluation, $Q(D,BS) = \{\{(a)\}\}$, whereas $Q'(D,BS) = \{\{(a),(a)\}\}$. Thus, database D is a counterexample to $Q \equiv_{\{\sigma\},BS} Q'$, which proves that the chase step $Q \Rightarrow^{\sigma}_{BS} Q'$ using σ is not sound. □

F. COUNTEREXAMPLE DATABASE FOR EXAMPLE 5.1

This section of the appendix provides the counterexample database for Example **??**. Database D is a counterexample to soundness of chase step $Q_4 \Rightarrow^{\sigma_1}_{B} Q_4^{(1)}$. In D, let the relations be as follows: $P = \{\{(1,2)\}\}$, $R = \emptyset$, $S = \{\{(1,3)\}\}$, $T = \{\{(1,4,5),\ (1,6,7)\}\}$, and $U = \emptyset$. Note that $D \models \Sigma$, for the set of dependencies Σ in Example **??**.

On this database D, the answer to Q_4 is $Q_4(D,B) = \{\{(1)\}\}$, whereas $Q_4^{(1)}(D,B) = \{\{(1),(1)\}\}$, by rules of bag semantics. From the fact that $Q_4(D,B)$ and $Q_4^{(1)}(D,B)$ are not the same *bags,* we conclude that the chase step $Q_4 \Rightarrow^{\sigma_1}_{B} Q_4^{(1)}$ is not sound under bag semantics.

G. UNIQUENESS THEOREMS FOR CHASE RESULTS

We begin this section of the appendix by formulating the version of Theorem 5.1 for the case of bag-set semantics.

THEOREM G.1. *Given a CQ query Q and a set Σ of embedded dependencies on database schema $\mathcal{D}$, such that there exists a chase result $(Q)_{\Sigma,S}$ for Q and Σ under* set *semantics. Then there exists a result $(Q)_{\Sigma,BS}$ of* sound *chase for Q and Σ under* bag-set *semantics, unique up to isomorphism of its canonical representation.*[14] *That is, for two sound-chase results $(Q)^{(1)}_{\Sigma,BS}$ and $(Q)^{(2)}_{\Sigma,BS}$ for Q and Σ, $(Q)^{(1)}_{\Sigma,BS} \equiv_{BS} (Q)^{(2)}_{\Sigma,BS}$ in the absence of dependencies.* □

We now provide a proof for Theorem 5.1. An adaptation of the proof to the statement of Theorem G.1 is straightforward.

PROOF. (Theorem 5.1) We first establish that, by the definition of soundness of the chase result $(Q)^{(1)}_{\Sigma,B}$, there exists a chase sequence C_1 using Σ, such that C_1 starts with Q and ends with $(Q)^{(1)}_{\Sigma,B}$, and such that all chase steps in C_1 are sound under bag semantics. Similarly, we establish that there exists a chase sequence C_2 using Σ, such that C_2 starts with Q and ends with $(Q)^{(2)}_{\Sigma,B}$, and such that all chase steps in C_2 are sound under bag semantics.

The proof of Theorem 5.1 is by contrapositive. Assume, toward contradiction, that $(Q)^{(1)}_{\Sigma,B}$ and $(Q)^{(2)}_{\Sigma,B}$ are not isomorphic after removal of duplicate subgoals that correspond to set-valued relations in the database schema $\mathcal{D}$. Let us denote by $(\bar{Q})^{(1)}_{\Sigma,B}$ the result of removing such "set-valued" duplicate subgoals from $(Q)^{(1)}_{\Sigma,B}$, and let us use the analogous notation $(\bar{Q})^{(2)}_{\Sigma,B}$ for $(Q)^{(2)}_{\Sigma,B}$.

Suppose, w.l.o.g., that $(\bar{Q})^{(1)}_{\Sigma,B}$ has a nonempty set of subgoals $p_1(\bar{X}_1),\ldots,p_m(\bar{X}_m)$ such that this set of subgoals does not have a counterpart in the image of any injective homomorphism from $(\bar{Q})^{(1)}_{\Sigma,B}$ to $(\bar{Q})^{(2)}_{\Sigma,B}$. It is clear that $p_1(\bar{X}_1),\ldots,p_m(\bar{X}_m)$ cannot be a subset of all the subgoals in the body of the original query Q. (By definition of sound chase steps, no chase steps using embedded dependencies ever remove original query subgoals.) Then, from the sound chase sequence C_1, we can form a sequence C_1' of sound chase steps that (i) uses a *subsequence* of the sequence of dependencies applied in C_1, and (ii) starts with Q and ends with adding all the subgoals in $p_1(\bar{X}_1),\ldots,p_m(\bar{X}_m)$ to Q. By definition of sound chase, there must exist a nonempty suffix subsequence C_1'' of C_1' such that all chase steps in C_1'' apply to the chase result $(Q)^{(2)}_{\Sigma,B}$ of chase sequence C_2, and such that applying the respective (to C_1'') dependencies in Σ to $(Q)^{(2)}_{\Sigma,B}$ would result in adding to $(Q)^{(2)}_{\Sigma,B}$ a set of subgoals that would be an image of $p_1(\bar{X}_1),\ldots,p_m(\bar{X}_m)$ in some injective homomorphism from $(\bar{Q})^{(1)}_{\Sigma,B}$ to $(\bar{Q})^{(2)}_{\Sigma,B}$. We thus arrive at a contradiction with the condition of Theorem 5.1, which states that $(Q)^{(2)}_{\Sigma,B}$ is a (terminal) result of sound chase for Q using Σ under bag semantics. (That is, the contradiction is with the assumption that no sound chase steps of the form $(Q)^{(2)}_{\Sigma,B} \Rightarrow^{\sigma}_{B} Q'$ are possible, where $\sigma \in \Sigma$.)

The case where some of $p_1(\bar{X}_1),\ldots,p_m(\bar{X}_m)$ were eliminated in $(Q)^{(2)}_{\Sigma,B}$ by use of one or more egds in Σ is analogous to the above tgd case, except that the contradiction in the case of egds is with our assumption that $(Q)^{(1)}_{\Sigma,B}$ is a (terminal) result of sound chase for Q using Σ under bag semantics. That is, those same egds can be applied to $(Q)^{(1)}_{\Sigma,B}$, hence $(Q)^{(1)}_{\Sigma,B}$ is not a result of sound chase under bag semantics. □

H. COMPLEXITY OF SOUND CHASE

In this section of the appendix, we establish for Theorem 5.2 the lower bound on the complexity of sound chase under each of bag and bag-set semantics, using sets of weakly acyclic dependencies.

H.1 Weakly Acyclic Dependencies

We provide here the definition and discussion of [11] for weakly acyclic dependencies.

The chase-termination property under set semantics is in general undecidable for CQ queries and dependencies given by tgds and egds. However, the notion of *weak acyclicity* of a set of dependencies is sufficient to guarantee that any chase sequence terminates. This is the least restrictive sufficient termination condition that has been generally studied in the literature (but see [10] for a generalization). The weak acyclicity condition appears to hold in all practical scenarios.

[14]See Theorem 2.1 in Section 2.3.

DEFINITION H.1. *(**Weakly acyclic set of dependencies**) Let Σ be a set of tgds over a fixed schema. Construct a directed graph, called the* dependency graph, *as follows: (1) there is a node for every pair (R, A), with R a relation symbol of the schema and A an attribute of R; call such pair (R, A) a* position; *(2) add edges as follows: for every tgd $\phi(\bar{X}) \rightarrow \exists \bar{Y}\ \psi(\bar{X}, \bar{Y})$ in Σ and for every X in $\bar{X}$ that occurs in ψ:*

For every occurrence of X in ϕ in position (R, A_i):

(a) for every occurrence of X in ψ in position (S, B_j), add an edge $(R, A_i) \rightarrow (S, B_j)$;

(b) in addition, for every existentially quantified variable Y and for every occurrence of Y in ψ in position (T, C_k), add a special edge $(R, A_i) \rightarrow^* (T, C_k)$.

Note that there may be two edges in the same direction between two nodes, if exactly one of the two edges is special. Then Σ is weakly acyclic *if the dependency graph has no cycle going through a special edge. We say that a set of tgds and egds is* weakly acyclic *if the set of all its tgds is weakly acyclic.* □

THEOREM H.1. *[12, 14] If Σ is a weakly acyclic set of tgds and egds, then the chase with Σ of any CQ query Q under set semantics terminates in finite time.* □

The complexity of the chase. For a fixed database schema and set Σ of dependencies, if Σ is weakly acyclic then under set semantics any chase sequence terminates in polynomial time in the size of the query being chased (as shown in [12, 14]). The fixed-size assumption about schemas and dependencies is often justified in practice, where one is usually interested in repeatedly reformulating incoming queries for the same setting with schemas and dependencies. Nonetheless, the degree of the polynomial depends on the size of the dependencies and care is needed to implement the chase efficiently. Successive implementations have shown that in practical situations the chase is eminently usable [11].

The complexity of reformulation under set semantics (in C&B). Assume that under set semantics the chase of any query with Σ terminates in polynomial time (for fixed database schema). Then checking whether a CQ query Q admits a reformulation is NP-complete in the size of Q. Checking whether a given query Q' is a Σ-minimal reformulation of Q is NP-complete in the sizes of Q and Q'. For arbitrary sets of dependencies (for which the chase may not even terminate), the above problems are undecidable.

H.2 The Lower Complexity Bound

We now establish for Theorem 5.2 the lower bound on the complexity of sound chase using weakly acyclic dependencies under each of bag and bag-set semantics, as follows.

EXAMPLE H.1. *On a database schema $\mathcal{D} = \{P_1,\ P_2,\ \ldots,\ P_m\}$ where each relation symbol has arity 2, consider a query Q with a single subgoal p_1:*

$Q(X, Y)\ :-\ p_1(X, Y).$

Suppose the database schema $\mathcal{D}$ satisfies a set Σ of tgds of the following form:

$\sigma^{(1)}_{i,j}\ :\ p_i(X, Y) \rightarrow\ \exists Z\ p_j(Z, X)$
$\sigma^{(2)}_{i,j}\ :\ p_i(X, Y) \rightarrow\ \exists W\ p_j(Y, W)$

Σ has one tgd $\sigma^{(1)}_{i,j}$ and one tgd $\sigma^{(2)}_{i,j}$ for each pair (i, j), where $i \in \{1, \ldots, m-1\}$ and $j \in \{i+1, \ldots, m\}$. Thus, the number of dependencies in Σ is quadratic in m.

We show one partial chase result (under set semantics) of the query Q under dependencies Σ, for $m \geq 2$:

$Q'(X, Y)\ :-\ p_1(X, Y),\ p_2(Z_1, X),\ p_2(Y, Z_2).$

Q' is the result of applying to Q tgds $\sigma^{(1)}_{1,2}$ and $\sigma^{(2)}_{1,2}$. Observe that Q' has a self-join of the relation P_2. □

For the terminal result $(Q)_{\Sigma,S}$ of chase of the query Q using the tgds Σ under set semantics in Example H.1, we can show that the size of $(Q)_{\Sigma,S}$ is exponential in the size of Q and Σ. Specifically, the size of $(Q)_{\Sigma,S}$ is exponential in the size m of the database schema $\mathcal{D}$. Intuitively, just as Q' has two subgoals for predicate p_2, the query $(Q)_{\Sigma,S}$ has two subgoals for p_2, four subgoals for p_3, and so on.

EXAMPLE H.2. *We continue Example H.1. We build a set Σ' of dependencies from the set Σ of Example H.1 by adding $3m$ functional dependencies (fds): For each $i \in \{1, \ldots, m\}$, we add the following three fds for the relation P_i in $\mathcal{D}$:*

$\sigma^{(1)}_i\ :\ p_i(X, Y)\ \wedge\ p_i(X, Z) \rightarrow\ Y = Z$
$\sigma^{(2)}_i\ :\ p_i(Y, X)\ \wedge\ p_i(Z, X) \rightarrow\ Y = Z$
$\sigma^{(3)}_i\ :\ p_i(X, Y, Z_1)\ \wedge\ p_i(X, Y, Z_2) \rightarrow\ Z_1 = Z_2$

That is, in all databases that satisfy the first two fds for i in Σ', the core-set of P_i does not have repeated values of either attribute. The third fd for P_i guarantees that relation P_i is set *valued in all instances of the database schema $\mathcal{D}$. Here, the* third *attribute of P_i is its* tuple-id *attribute. Please see Appendix C for the details on using egds for enforcing set-valuedness of relations in all instances of a given database schema.*

Note that the addition of these fds transforms the tgds Σ of Example H.1 into key-based tgds Σ' (see Definition 5.1). Thus, for the terminal result $(Q)_{\Sigma',B}$ of sound chase of the query Q under the dependencies Σ' under bag semantics, the size of $(Q)_{\Sigma',B}$ is exponential in the size of Q and Σ'. The same relationship holds under bag-set semantics between the size of $(Q)_{\Sigma',BS}$ and the sizes of Q and Σ. □

By the results of Section 4, chase of CQ query Q under key-based tgds Σ results in a query that is equivalent to Q under Σ under each of bag and bag-set semantics for query evaluation. Observing that the dependencies Σ' of Example H.2 are weakly acyclic (and, in fact, strictly acyclic), completes the construction of the infinite family of pairs (Q, Σ'), one pair for each natural-number value of m, such that the size of each of $(Q)_{\Sigma,B}$ and $(Q)_{\Sigma,BS}$ (both constructed using sound chase) is polynomial in the size of Q and exponential in size of Σ.

I. SATISFIABLE DEPENDENCIES ARE QUERY BASED

In this section of the appendix we provide Theorem I.1, which is the analog of Theorem 5.3 for the case of bag-set semantics. We then supply a proof of Theorem 5.3; the proof of Theorem I.1 is similar. Finally, we outline the counterpart of algorithm MAX-BAG-Σ-SUBSET (of Section 5.3) for the case of bag-set semantics.

THEOREM I.1. **(Unique $\Sigma^{max}_{BS}(Q, \Sigma) \subseteq \Sigma$)** *Given a CQ query Q and set Σ of embedded dependencies, such that there exists a* set-*chase result $(Q)_{\Sigma,S}$ for Q and Σ. Let Q_n be the result of sound chase for Q and Σ under* bag-set *semantics, with canonical database $D^{(Q_n)}$. Then there exists a unique subset $\Sigma^{max}_{BS}(Q, \Sigma)$ of Σ, such that:*

- $D^{(Q_n)} \models \Sigma^{max}_{BS}(Q, \Sigma)$*, and*
- *for each proper superset Σ' of $\Sigma^{max}_{BS}(Q, \Sigma)$ such that $\Sigma' \subseteq \Sigma$, $D^{(Q_n)} \models \Sigma'$ does* not *hold.* $\square$

We now turn to the proof of Theorem 5.3. We first observe that the process of sound chase of a CQ query using a set Σ of embedded dependencies under bag semantics can be modeled as state transitions for Σ, with certain conditions on the final state, which corresponds to obtaining the result of the chase. The termination conditions are formalized in Proposition I.1; we first set up the terminology required to formulate Proposition I.1.

Suppose we are given a CQ query Q and a finite set Σ of embedded dependencies, such that there exists a *set*-chase result $(Q)_{\Sigma,S}$ for Q and Σ. Consider an arbitrary chase sequence $\mathbf{C} = Q_0, Q_1, \ldots$, such that (i) $Q_0 = Q$, and (ii) every query Q_{i+1} $(i \geq 0)$ in $\mathbf{C}$ is obtained from Q_i by a sound chase step $Q_i \Rightarrow^{\sigma}_{B} Q_{i+1}$ using a dependency $\sigma \in \Sigma$. By Proposition 5.1, the chase sequence $\mathbf{C}$ is finite, that is, $\mathbf{C} = Q_0, Q_1, \ldots, Q_n$, such that $n \in \mathbf{N} \cup \{0\}$ and such that query $Q_n = (Q)_{\Sigma,B}$. Moreover, by Theorem 5.1 we have that the query Q_n is bag-equivalent in the absence of dependencies[15] to the terminal queries in all sound-chase sequences for Q and Σ under bag semantics.

Given a chase sequence $\mathbf{C}$ as defined above, with chase result $Q_n = (Q)_{\Sigma,B}$, we assign a unique ID to each subgoal of Q_n. We then "propagate the IDs back" to all the queries in $\mathbf{C}$, so that the enumeration of the subgoals is consistent across all the elements of $\mathbf{C}$. If extra subgoals are encountered in non-terminal elements of $\mathbf{C}$, we assign unique IDs to those subgoals as well. (The only case when a query Q_i, $i < n$, in $\mathbf{C}$ could have an extra subgoal compared to Q_n is when the procedure of dropping duplicate subgoals has been applied to either Q_i or its successors in $\mathbf{C}$. See Theorems 2.1, 4.1, and 4.2.) In what follows, we refer to the jth subgoal of query Q_i as $s^{(i)}_j$.

Fix an arbitrary $i \in \{0, \ldots, n\}$, and consider query Q_i in the chase sequence $\mathbf{C}$. Given an arbitrary dependency $\sigma \in \Sigma$, of the form $\sigma : \phi(\bar{U}, \bar{W}) \rightarrow \exists \bar{V}\ \psi(\bar{U}, \bar{V})$, we define the state of σ w.r.t. Q_i in $\mathbf{C}$ as follows:

- Dependency σ is *pre-applicable* to Q_i if the chase of none of $Q_0, Q_1, \ldots, Q_i$ with σ is applicable; that is, for each $j \in \{0, \ldots, i\}$, there does not exist a homomorphism from the left-hand side ϕ of σ to the body of the query Q_j.
- Dependency σ is *soundly applicable* to set of subgoals $S = \{s^{(i)}_{j1}, \ldots, s^{(i)}_{jk}\}$ of query Q_i, for some $k > 0$, if there exists a proper subset θ, of size $k' \geq k$, of $\phi \wedge \psi$ (of σ), with the following properties:
  - θ is a superset of ϕ;
  - there exists a homomorphism h from θ to exactly the subgoals $s^{(i)}_{j1}, \ldots, s^{(i)}_{jk}$ of query Q_i, such that h cannot be extended to a homomorphism from $\phi \wedge \psi$ to the body of the query Q_i (see Section 2.4 for further details on this definition); and
  - chase step $Q_i \Rightarrow^{\sigma}_{B} Q'$, where Q' is a CQ query, is sound; that is, $Q' \equiv_{\Sigma,B} Q_i$.
- Dependency σ is *unsoundly applicable* to set of subgoals $S = \{s^{(i)}_{j1}, \ldots, s^{(i)}_{jk}\}$ of query Q_i, for some $k > 0$, if there exists a proper subset θ, of size $k' \geq k$, of $\phi \wedge \psi$ (of σ), with the following properties:
  - θ is a superset of ϕ;
  - there exists a homomorphism h from θ to exactly the subgoals $s^{(i)}_{j1}, \ldots, s^{(i)}_{jk}$ of query Q_i, such that h cannot be extended to a homomorphism from $\phi \wedge \psi$ to the body of the query Q_i (see Section 2.4 for further details on this definition); and
  - chase step $Q_i \Rightarrow^{\sigma}_{B} Q'$, where Q' is a CQ query, is *unsound;* that is, $Q' \equiv_{\Sigma,B} Q_i$ does not hold.
- Finally, dependency σ is *post-applicable* to Q_i (assuming $i > 0$) if (a) σ is neither soundly applicable nor unsoundly applicable to Q_i, and (b) there exists a $j \in \{0, \ldots, i-1\}$ such that σ has been used in a sound chase step $Q_j \Rightarrow^{\sigma}_{B} Q_{j+1}$. Observe that in this case, by definition of (sound) chase steps there exists a homomorphism from the conjunction of the left-hand side ϕ of σ with the right-hand side ψ of σ to the body of the query Q_i.

In the above definition of the state of $\sigma \in \Sigma$ w.r.t. Q_i in $\mathbf{C}$, the only difference between the states "soundly applicable" and "unsoundly applicable" is the soundness property of the chase step in question. Specifically, in the state "σ is soundly applicable to Q_i" the chase step $Q_i \Rightarrow^{\sigma}_{B} Q'$ is sound under bag semantics, whereas in the state "σ is unsoundly applicable to Q_i", the chase step $Q_i \Rightarrow^{\sigma}_{B} Q'$ is unsound.

We now define the *state of the set of embedded dependencies Σ w.r.t. Q_i in* $\mathbf{C}$, as a total mapping $s^{\mathbf{C}}_i$ from Σ to the set of the four above states (pre-applicable, post-applicable, soundly-applicable, and unsoundly-applicable), where the state $s^{\mathbf{C}}_i(\sigma)$ of each $\sigma \in \Sigma$ w.r.t. Q_i in $\mathbf{C}$ is as follows:

- $s^{\mathbf{C}}_i(\sigma)$ = "soundly-applicable" if and only if there exists a set S of subgoals of Q_i such that σ is soundly applicable to S in Q_i;
- $s^{\mathbf{C}}_i(\sigma)$ = "unsoundly-applicable" if and only if there exists *no* subset S of subgoals of Q_i such that σ is soundly applicable to S in Q_i, *and* there exists a set S' of subgoals of Q_i such that σ is unsoundly applicable to S' in Q_i;
- $s^{\mathbf{C}}_i(\sigma)$ = "post-applicable" if σ and Q_i satisfy the conditions (a) and (b) of post-applicability, see above; and
- $s^{\mathbf{C}}_i(\sigma)$ = "pre-applicable" if σ and Q_i satisfy the conditions of pre-applicability, see above.

We now establish straightforward facts about the states of Σ w.r.t. particular queries in the sound-chase sequence $\mathbf{C} = Q_0, \ldots, Q_n$, where Q_n is the result of the

[15] Other than the set-enforcing dependencies on stored relations, see Theorem 5.1.

sound chase of Q using Σ under bag semantics. The proofs of all the claims in Proposition I.1 are immediate from the definitions in this section of the appendix and from the definitions of chase steps, see Section 2.4.

PROPOSITION I.1. *For a CQ query Q and a set of embedded dependencies Σ such that there exists a set-chase result $(Q)_{\Sigma,S}$ for Q and Σ. Let $\mathbf{C} = Q_0, \ldots, Q_n$ be a sound-chase sequence for Q and Σ under bag semantics. In $\mathbf{C}$, $Q_0 = Q$, and Q_n is the result $(Q)_{\Sigma,B}$ of the sound chase of Q using Σ under bag semantics. Then the following holds about the states of Σ w.r.t. queries in $\mathbf{C}$.*

1. *Suppose that in the state $s_0^{\mathbf{C}}$ of Σ w.r.t. query Q_0 in chase sequence $\mathbf{C}$, for all $\sigma \in \Sigma$ it holds that $s_0^{\mathbf{C}}(\sigma)$ is either "pre-applicable" or "unsoundly-applicable". Then $\mathbf{C} = Q_0$. That is, Q is isomorphic to $(Q)_{\Sigma,B}$.*
2. *Consider the state $s_n^{\mathbf{C}}$ of Σ w.r.t. query Q_n in chase sequence $\mathbf{C}$. Then for all $\sigma \in \Sigma$ it must hold that $s_n^{\mathbf{C}}(\sigma)$ is one of "pre-applicable", "post-applicable", and "unsoundly-applicable".*
3. *For an arbitrary $i \in \{0, \ldots, n-1\}$ (assuming $n > 0$), consider the state $s_i^{\mathbf{C}}$ of Σ w.r.t. query Q_i and the state $s_{i+1}^{\mathbf{C}}$ of Σ w.r.t. query Q_{i+1}. Then*
   (a) *there must exist a $\sigma^* \in \Sigma$ such that $s_i^{\mathbf{C}}(\sigma^*)$ is "soundly applicable", and*
   (b) *for each $\sigma \in (\Sigma - \{\sigma^*\})$, $s_i^{\mathbf{C}}(\sigma) = s_{i+1}^{\mathbf{C}}(\sigma)$.* □

We are now ready to prove Theorem 5.3.

PROOF. (Theorem 5.3) Consider a fixed pair (Q, Σ) satisfying the conditions of Theorem 5.3, and let Q_n be the result of sound chase for Q and Σ under bag semantics, with canonical database $D^{(Q_n)}$. We show that the set $\Sigma_B^{max}(Q,\Sigma)$ is the result of removing from Σ exactly those tgds σ such that the chase step $Q_n \Rightarrow_B^{\sigma} Q'$, with some CQ query Q' being the outcome of the chase step, is unsound under bag semantics. This claim is, in fact, immediate from Proposition I.1, in which it is shown that, for each dependency σ in Σ such that σ is applicable to Q_n, σ is *unsoundly* applicable to Q_n. □

Finally, we outline the counterpart of algorithm MAX-BAG-Σ-SUBSET (of Section 5.3) for the case of bag-set semantics.

```
Algorithm 2: Max-Bag-Set-Σ-Subset($Q, \Sigma$)
Input  : CQ query $Q$, set $\Sigma$ of embedded dependencies
         such that chase result $(Q)_{\Sigma,S}$ exists
Output : $\Sigma_{BS}^{max}(Q,\Sigma) \subseteq \Sigma$ s. t.
         (1) $D^{((Q)_{\Sigma,BS})} \models \Sigma_{BS}^{max}(Q,\Sigma)$, and
         (2) $\forall$ $\Sigma'$ such that $\Sigma_{BS}^{max}(Q,\Sigma) \subset \Sigma' \subseteq \Sigma$,
         $D^{((Q)_{\Sigma,BS})} \not\models \Sigma'$
1. $(Q)_{\Sigma,BS}$ := $soundChase(BS, Q, \Sigma)$;
2. $\Sigma_{BS}^{max}(Q,\Sigma)$ := $\Sigma$;
3. for each $\sigma$ in $\Sigma$ do
    4. if $soundChaseStep(\sigma, BS, (Q)_{\Sigma,BS}) = false$
    then
        5. $\Sigma_{BS}^{max}(Q,\Sigma)$ := $\Sigma_{BS}^{max}(Q,\Sigma) - \{\sigma\}$;
6. return $\Sigma_{BS}^{max}(Q,\Sigma)$;
```

The correctness and complexity results for MAX-BAG-SET-Σ-SUBSET are the same as their counterparts for algorithm MAX-BAG-Σ-SUBSET, see Theorem 5.4 and Section 5.3 for the details.

J. PROOFS OF Σ-EQUIVALENCE-TESTS FOR CQ QUERIES

To prove Theorems 6.1 and Theorem 6.2, we first make a straightforward observation, as follows.

PROPOSITION J.1. *Given two queries Q and Q' and a set of embedded dependencies Σ. Let X be one of B, BS, S, which stand for bag, bag-set, and set semantics, respectively. Then $Q \equiv_X Q'$ implies $Q \equiv_{\Sigma,X} Q'$.* □

The proof of Proposition J.1 is straightforward from the definitions of query equivalence in presence and in the absence of dependencies.

The proof of Theorem 6.1 is immediate from Propositions 5.1 and J.1, from Theorem 5.1, and from Lemmas J.1 and J.2. Similarly, the proof of Theorem 6.2 is immediate from Propositions 5.1 and J.1, from the analog of Theorem 5.1 for bag-set semantics (see Theorem G.1), and from straightforward analogs of Lemmas J.1 and J.2 for the case of bag-set semantics for query evaluation.

LEMMA J.1. *Given CQ queries Q and Q', and given a set of embedded dependencies Σ on schema $\mathcal{D}$ such that there exist* set-*chase results $(Q)_{\Sigma,S}$ for Q and $(Q')_{\Sigma,S}$ for Q'. Then $Q \equiv_{\Sigma,B} Q'$ implies $(Q)_{\Sigma,B} \equiv_B (Q')_{\Sigma,B}$ in the absence of all dependencies other than the set-enforcing dependencies on $\mathcal{D}$.* □

PROOF. First, from Proposition 5.1 we obtain that sound chase of each of Q and Q' using Σ is guaranteed to terminate under bag semantics. Further, from Theorem 5.1 it follows that there exist (1) a unique result $(Q)_{\Sigma,B}$ of sound chase for Q, and (2) a unique result $(Q')_{\Sigma,B}$ of sound chase for Q' . Both results are unique in the absence of all dependencies other than the set-enforcing dependencies on $\mathcal{D}$, call these set-enforcing dependencies $\Sigma' \subseteq \Sigma$.

From $Q \equiv_{\Sigma,B} Q'$ and by the soundness of chase in obtaining $(Q)_{\Sigma,B}$ and $(Q')_{\Sigma,B}$, we have $(Q)_{\Sigma,B} \equiv_{\Sigma,B} (Q')_{\Sigma,B}$. That is, on each bag-valued database D that satisfies Σ, we have that $Q(D,B)$ and $Q'(D,B)$ are the same as bags.

To show that $(Q)_{\Sigma,B} \equiv_B (Q')_{\Sigma,B}$ in the absence of all dependencies other than Σ', it remains to prove that $Q(D,B)$ and $Q'(D,B)$ are also the same as bags on each database D that does not satisfy Σ but does satisfy Σ'. There are two cases:

Case 1: Suppose D violates only those dependencies that are not relevant in sound chase to either Q or Q'. (In the terminology of Section I, those would be exactly the dependencies that are pre-applicable to each of $(Q)_{\Sigma,B}$ and $(Q')_{\Sigma,B}$.) In this case, D does not violate any dependencies as far as $(Q)_{\Sigma,B}$ or $(Q')_{\Sigma,B}$ are concerned, as formalized in Theorem 5.3. Thus from $(Q)_{\Sigma,B} \equiv_{\Sigma,B} (Q')_{\Sigma,B}$ we obtain that $Q(D,B)$ and $Q'(D,B)$ are the same as bags on D.

Case 2: Suppose D violates at least one dependency that is relevant in *sound* chase to either Q or Q'. (In the terminology of Section I, those would be exactly the dependencies that are post-applicable to each of $(Q)_{\Sigma,B}$ and $(Q')_{\Sigma,B}$.) Still, by Theorem 5.3 the definitions of

$(Q)_{\Sigma,B}$ and of $(Q')_{\Sigma,B}$ ensure that all such relevant dependencies are enforced (i.e., do not fail) on all assignments γ that satisfy each of $(Q)_{\Sigma,B}$ and $(Q')_{\Sigma,B}$ w.r.t. D. Let D_Q be the union of all tuples in all such satisfying assignments for $(Q)_{\Sigma,B}$ w.r.t D; $D_{Q'}$ is defined analogously for $(Q')_{\Sigma,B}$. Then $D' = D_Q \bigcup D_{Q'}$ satisfies all the dependencies of Σ that are relevant in chase to either Q or Q'. Thus, from $(Q)_{\Sigma,B} \equiv_{\Sigma,B} (Q')_{\Sigma,B}$ we obtain that $Q(D', B)$ and $Q'(D', B)$ are the same as bags. From the fact that none of the tuples of D that are not in D' participates in forming either $Q(D, B)$ or $Q'(D, B)$, it follows that $Q(D, B)$ and $Q'(D, B)$ are the same as bags *on database* D. $\square$

Lemma J.2. *Given CQ queries Q, Q', and given embedded dependencies Σ on schema $\mathcal{D}$ such that there exist* set-*chase results $(Q)_{\Sigma,S}$ for Q and $(Q')_{\Sigma,S}$ for Q'. Then $Q \equiv_{\Sigma,B} Q'$ holds whenever $(Q)_{\Sigma,B} \equiv_B (Q')_{\Sigma,B}$ in the absence of all dependencies other than the set-enforcing dependencies on $\mathcal{D}$.* $\square$

The proof of Lemma J.2 is immediate from the fact that each of $(Q)_{\Sigma,B}$ and $(Q')_{\Sigma,B}$ was obtained using sound chase steps under bag semantics (which implies $(Q)_{\Sigma,B} \equiv_{\Sigma,B} Q$ and $(Q')_{\Sigma,B} \equiv_{\Sigma,B} Q'$), as well as from Propositions 5.1 and J.1 and from transitivity of bag equivalence in presence of dependencies.

K. Σ-BASED VERSION OF PROP. 2.1

In this appendix we provide the proof of Proposition 6.1, which is the dependency-based version of Proposition 2.1 ([4], see Section 2.3 of this current paper). By Theorems 6.1 and 6.2, the proof works both for the formulation of Proposition 6.1 and for the formulation that parallels Proposition 2.1 (see Proposition K.1 below.) We also provide a proof of Proposition 6.2. Finally, we provide the analogs of Theorem 6.4 for (a) CQ queries under bag-set semantics, and for (b) CQ queries with grouping and aggregation.

Proof. (Proposition 6.1)

Proof of (1): Assume

$$Q \equiv_{\Sigma,B} Q'. \tag{18}$$

or, equivalently (by Theorem 6.1), assume

$$(Q)_{\Sigma,B} \equiv_B (Q')_{\Sigma,B} \tag{19}$$

in the absence of all dependencies other than the set-enforcing dependencies of the given database schema. Then Equation 20

$$(Q)_{\Sigma,B} \equiv_{BS} (Q')_{\Sigma,B}. \tag{20}$$

follows from Equation 19 by Proposition 2.1. Equation 21

$$(Q)_{\Sigma,B} \equiv_{\Sigma,BS} (Q')_{\Sigma,B}. \tag{21}$$

follows from Equation 20 by Proposition J.1. Equation 22

$$((Q)_{\Sigma,B})_{\Sigma,BS} \equiv_{BS} ((Q')_{\Sigma,B})_{\Sigma,BS}. \tag{22}$$

follows from Equation 21 by Theorem 6.2. Equation 23

$$(Q)_{\Sigma,BS} \equiv_{BS} (Q')_{\Sigma,BS}. \tag{23}$$

follows from Equation 22 for the following reasons:

- By Proposition 5.2 (also see Theorem 4.1 and the definitions of chase steps), the set $\Sigma_1 \subseteq \Sigma$ of dependencies that are soundly applicable to a query under bag semantics is a subset of the set $\Sigma_2 \subseteq \Sigma$ of dependencies that are soundly applicable to the same query under bag-set semantics.
- From Theorem 5.1 and its analog for bag-set semantics (Theorem G.1), it follows that $((Q)_{\Sigma,B})_{\Sigma,BS} \equiv_{BS} (Q)_{\Sigma,BS}$, and similarly $((Q')_{\Sigma,B})_{\Sigma,BS} \equiv_{BS} (Q')_{\Sigma,BS}$.
- By transitivity of $\equiv_{BS}$, we obtain Equation 23.

Finally, Equation 24

$$Q \equiv_{\Sigma,BS} Q'. \tag{24}$$

follows from Equation 23 by Theorem 6.2.

Proof of (2): Assume

$$Q \equiv_{\Sigma,BS} Q'. \tag{25}$$

or, equivalently (by Theorem 6.2), assume

$$(Q)_{\Sigma,BS} \equiv_{BS} (Q')_{\Sigma,BS}. \tag{26}$$

Then Equation 27

$$(Q)_{\Sigma,BS} \equiv_S (Q')_{\Sigma,BS}. \tag{27}$$

follows from Equation 26 by Proposition 2.1. Equation 28

$$(Q)_{\Sigma,BS} \equiv_{\Sigma,S} (Q')_{\Sigma,BS}. \tag{28}$$

follows from Equation 27 by Proposition J.1. Equation 29

$$((Q)_{\Sigma,BS})_{\Sigma,S} \equiv_S ((Q')_{\Sigma,BS})_{\Sigma,S}. \tag{29}$$

follows from Equation 28 by Theorem 2.2. Equation 30

$$(Q)_{\Sigma,S} \equiv_S (Q')_{\Sigma,S}. \tag{30}$$

follows from Equation 29 for the following reasons:

- By Proposition 5.2 (also see Theorem 4.3 and the definitions of chase steps), the set $\Sigma_1 \subseteq \Sigma$ of dependencies that are soundly applicable to a query under bag-set semantics is a subset of the set $\Sigma_2 \subseteq \Sigma$ of dependencies that are (always soundly) applicable to the same query under set semantics.
- From the analog of Theorem 5.1 for bag-set semantics (Theorem G.1) and from the definitions of chase steps, it follows that $((Q)_{\Sigma,BS})_{\Sigma,S} \equiv_S (Q)_{\Sigma,S}$, and similarly $((Q')_{\Sigma,BS})_{\Sigma,S} \equiv_S (Q')_{\Sigma,S}$.
- By transitivity of $\equiv_S$, we obtain Equation 30.

Finally, Equation 31

$$Q \equiv_{\Sigma,S} Q'. \tag{31}$$

follows from Equation 30 by Theorem 2.2. $\square$

Proposition K.1. *Given two CQ queries Q_1 and Q_2, and a set of embedded dependencies Σ, such that there exists the* set-*chase result in chase of each of Q_1 and Q_2 using Σ. Then (1) $Q_1 \equiv_{\Sigma,B} Q_2$ implies $Q_1 \equiv_{\Sigma,BS} Q_2$, and (2) $Q_1 \equiv_{\Sigma,BS} Q_2$ implies $Q_1 \equiv_{\Sigma,S} Q_2$.* $\square$

We next provide a proof of Proposition 6.2.

PROOF. (Proposition 6.2) Consider a pair (Q, Σ) that satisfies conditions of Theorem 5.3. By definition of chase steps (see Section 2.4), in an arbitrary *set*-chase sequence $\mathbf{C} = Q, Q_1, \ldots$ for Q and Σ, for each element Q_i of $\mathbf{C}$ such that Q_{i+1} is also an element of $\mathbf{C}$, it holds that $Q_{i+1} \sqsubseteq_S Q_i$ in the absence of dependencies. (Also, trivially, for each CQ query Q it holds that $Q \sqsubseteq_S Q$.) By transitivity and reflexivity of $\sqsubseteq_S$, for an arbitrary pair (Q_i, Q_{i+j}) (for $j \geq 0$) of elements of $\mathbf{C}$, it holds that $Q_{i+j} \sqsubseteq_S Q_i$. By definition of sound chase under bag and bag-set semantics (see Section 4), the same *set-containment* relationship $Q_{i+j} \sqsubseteq_S Q_i$ holds for an arbitrary pair (Q_i, Q_{i+j}) (for $j \geq 0$) of elements of a sound-chase sequence $\mathbf{C}'$ under bag or bag-set semantics. The rest of the proof of Proposition 6.2 is immediate from the result of Proposition 5.2 that $\Sigma_B^{max}(Q, \Sigma) \subseteq \Sigma_{BS}^{max}(Q, \Sigma) \subseteq \Sigma$ for the above fixed pair (Q, Σ) and from Proposition 6.1. $\square$

We now provide the analog of Theorem 6.4 for CQ queries under bag-set semantics.

THEOREM K.1. *Given CQ query Q and set Σ of embedded dependencies such that* set *chase of Q under Σ terminates in finite time. Then* BAG-SET-C&B *returns all Σ-minimal reformulations Q' such that $Q' \equiv_{\Sigma,BS} Q$.* $\square$

Finally, we provide the analog of Theorem 6.4 for CQ queries with grouping and aggregation.

THEOREM K.2. *Given CQ query Q with aggregate function max, min, sum, or count, and set Σ of embedded dependencies such that* set *chase of the core of Q under Σ terminates in finite time. Then (1) If the aggregate function of Q is max or min, then* MAX-MIN-C&B *returns all Σ-minimal reformulations Q' of Q such that $Q' \equiv_\Sigma Q$; (2) If the aggregate function of Q is sum or count, then* SUM-COUNT-C&B *returns all Σ-minimal reformulations Q' of Q such that $Q' \equiv_\Sigma Q$.* $\square$